VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2021 Estera Horváthová

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

VIRTUALIZAČNÍ PLATFORMA PRO BEZPEČNOSTNÍ CVIČENÍ
VIRTUALIZATION PLATFORM FOR CYBER RANGE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Estera Horváthová**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Tomáš Lieskovan**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Studentka: Estera Horváthová ***ID:*** 203407

Ročník: 3 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Virtualizační platforma pro bezpečnostní cvičení

POKYNY PRO VYPRACOVÁNÍ:

Hlavním cílem bakalářské práce je navrhnout a implementovat vhodné prostředí pro provedení bezpečnostních cvičení např. Red/Blue (tzv. cyber range). V teoretické části práce je hlavním cílem analyzovat dostupné virtualizační nástroje a jejich rozhraní k realizaci bezpečnostních cvičení. Analýzu zaměřte primárně na platformu OpenStack, její varianty a možnosti instalace (minimální hardwarové požadavky, varianty řešení př. Kolla, možné instalace př. All-In-One nebo více serverové atd.). Porovnejte výhody, nevýhody a nároky na zprovoznění jednotlivých variant. Na experimentálním pracovišti zprovozněte Vámi vybranou virtualizační platformu, platforma musí využívat OpenStack a být kompatibilní s KYPO projektem. Hlavním cílem praktické části bakalářské práce je vytvoření nejméně dvou bezpečnostních her na realizované virtualizační platformě. Vytvořené hry budou různé obtížnosti a budou využívat konceptu „hra o vlajku“. Hry budou plně využívat systémy OpenStack a KYPO.

DOPORUČENÁ LITERATURA:

[1] VIGNA, Giovanni. Teaching network security through live exercises. In: Security education and critical infrastructures. Springer, Boston, MA, 2003. p. 3-18.

[2] SIMPSON, Michael T.; BACKMAN, Kent; CORLEY, James. Hands-on ethical hacking and network defense. Cengage Learning, 2010.

Termín zadání: 1.2.2021 ***Termín odevzdání:*** 31.5.2021

Vedoucí práce: Ing. Tomáš Lieskovan

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Bakalářská práce se zabývá vytvořením a zprovozněním sandboxů pro vybrané scénáře pomocí nástroje sandbox-creator, který je kompatibilní s KYPO projektem a využívá platformu OpenStack. Teoretická část popisuje technologie a nástroje využívané pro vytváření tréningových platforem. V rámci technologií jsou popsány virtualizace, kontejnerizace a cloud computing a v rámci nástrojů jsou popsány VirtualBox, Docker nebo OpenStack. Práce dále popisuje vybrané cyber range platformy, které tyto technologie využívají. Praktická část popisuje kompatibilitu vybrané platformy s KYPO projektem, využívané konfigurační nástroje, přípravu experimentálního prostředí, vytváření a zprovoznění sandboxů pro laboratorní cvičení, která jsou uvedena v příloze.

KLÍČOVÁ SLOVA

cyber range, virtualizace, kontejnerizace, cloud computing, KYPO, OpenStack, Vagrant, Ansible, sandbox-creator, SSL-Strip, DDoS

ABSTRACT

The bachelor thesis deals with the creation and operation of sandboxes for selected scenarios using the sandbox-creator tool, which is compatible with the KYPO project and uses the OpenStack platform. The theoretical section describes the technologies and tools used to create training platforms. Virtualization, containerization and cloud computing are described in the technologies and VirtualBox is described in the tools, Docker or OpenStack. The work also describes selected cyber range platforms that use these technologies. The practical part describes the compatibility of the selected platform with the KYPO project, the configuration tools used, the preparation of the experimental environment, the creation and operation of sandboxes for laboratory exercises, which are listed in the annex.

KEYWORDS

cyber range, virtualization, containerization, cloud computing, KYPO, OpenStack, Vagrant, Ansible, sandbox-creator, SSL-Strip, DDoS

HORVÁTHOVÁ, Estera. *Virtualizační platforma pro bezpečnostní cvičení*. Brno, 2021, 79 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Tomáš Lieskovan,

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Virtualizační platforma pro bezpečnostní cvičení“ jsem vypracovala samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autorka uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušila autorská práva třetích osob, zejména jsem nezasáhla nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědoma následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autorky

PODĚKOVÁNÍ

Především děkuji svým bratrancům Ing. Tiboru Mátisovi a Ing. Miroslavu Matouškovi za motivaci a vzor, abych studovala na VUT Brno. Dále děkuji svým bývalým učitelům panu Ing. Závodnému a panu Mgr. Kovaříkovi ze SPŠ Břeclav, za cenný základ v oblasti webových stránek. Nakonec upřímně děkuji svému vedoucímu práce, panu Ing. Lieskovanovi, za pomoc, a to konkrétně: v zprovoznění v zobrazení literatury v Latexu, za doporučení přidání záznamu webové stránky do souboru */etc/hosts* při vytváření sandboxu sslstrip a za ukázku KYPO platformy na moji žádost.

Obsah

Seznam obrázků

Seznam výpisů

Úvod

Žijeme v době, kdy většina služeb je poskytována virtuálně. Služba je poskytována softwarem a pro své nasazení potřebuje hardwarové zdroje, tj. výpočetní paměť, procesor, úložiště, a síťové zdroje, které jsou souhrnně nazývány jako infrastruktura.

Z důvodu snížení spotřeby a efektivního využití fyzických hardwarových zdrojů se přešlo k nasazování více služeb za pomocí virtualizace na jeden fyzický hardwarový zdroj. Dále infrastrukturu bylo nutné zabezpečit a také průběžně testovat její odolnost vůči bezpečnostním incidentům, tj. útokům.

Za cílem zabezpečení bylo nutno vytvořit systém, pomocí kterého se výše zmíněným incidentům lze vyhnout. Systém, který připraví omezení a obranu vůči útokům. Jedná se o tréningové platformy, tzv. cyber range, které simulují vybranou nebo smyšlenou infrastrukturu pro testování a trénování vůči útokům. Zaměřují se na rozsáhlou část ochrany infrastruktury od obrany sítě až po penetrační testování, tj. "posouzení úrovně bezpečnostních mechanismů za pomocí simulace reálných situací, které by mohl použít útočník"[1].

Cyber range jsou využity u soutěže Hra o vlajku (Capture The Flag), kde existují tři typy. První typ, Jeopardy-style, je využíván většinou jednotlivci, kteří si chtějí zlepšit schopnosti v oblasti bezpečnosti. Druhý typ, Attack-Defend, je určen pro týmy a slouží pro vyzkoušení role útočníka a role obránce systému. Posledním typem je King of the hill, kde je cílem týmů získat co největší část infrastruktury protivníka.

Jak bylo zmíněné výše, cyber range je využívá simulaci, ke které jsou využity technologie jako virtualizace na úrovni hardwaru (virtualizace), virtualizace na úrovni operačního systému (kontejnerizace) a cloud computing pro virtualizaci rozsáhlejší infrastruktury. Tyto technologie využívají softwary jako jsou VirtualBox pro virtualizaci, Docker pro kontejnerizaci a OpenStack pro cloud computing.

Bakalářská práce je zaměřena na virtualizační technologie pro vytváření tréningových platforem. V kapitole 1 bude popsána základní charakteristika tréningových platforem. V následující kapitole 2 budou popsány právě technologie využívané pro tvorbu cyber range. Dále budou uvedeny vybrané virtualizační nástroje v rámci jednotlivých technologií. V kapitole 3 budou stručně popsány vybrané open-source platformy využívající popsané technologie. V rámci praktické části v kapitole 4 bude popsána souvislost zvolené platformy s KYPO projektem a budou uvedeny využité konfigurační nástroje a jejich krátká charakteristika používání. Pak bude následovat popis přípravy experimentálního prostředí pro realizaci praktické části práce. V další kapitole 5 bude popsáno vytvoření sandboxů pro realizaci vybraných útoků pro laboratorní cvičení. V přílohách A a B budou připojeny vytvořené laboratorní cvičení pro vytvořené prostředí.

1 Základní popis tréningových platforem

Z důvodu stále častějších útoků na nedostatečně zabezpečenou organizační infrastrukturu bylo nutno vytvořit systém, pomocí kterého se lze připravit na případný bezpečnostní incident, který může způsobit například ztráty nebo poškození citlivých dat. Systém, který simuluje reálné prostředí a využívá k tomu níže zmíněné technologie. Systém, který nabízí více druhů přípravy vůči útokům, například pro jednotlivce nebo pro týmy. Dále pro přípravu vůči těmto útokům jsou určeny i role pro účastníky.

1.1 Typy tréningových platforem

Platforma pro tréning, cyber range, je simulované znázornění organizační struktury z pohledu komunikační sítě, operačních systémů, nástrojů a aplikací [2]. Cyber range slouží pro testování a trénování dovedností na reakci vůči útokům. Je určena jak pro profesionály, tak pro studenty zaujímající se o oblast bezpečnosti. Trénovací platforma se zaměřuje hlavně na obranu sítě, detekce a mitigace (zmírnění) útoků, penetrační testování, analýzu malwaru, bezpečné programování a mnoho dalšího.

Cyber range je využíván na konferencích, cvičeních, ve formě soutěže Capture the Flag (CTF). Kde existují tři hlavní typy CTF:

- První typ, **Jeopardy-style**, je spíše určen pro samostatné hráče, kde jsou úkoly rozděleny do několika kategorií, jako například bezpečnost, kryptografie a steganografie, která slouží k ukrytí přenášené zprávy pomocí šifrování. Na začátku soutěže je spuštěn časovač určující dobu celé hry. V závislosti na počtu vyřešených úkolů se zvyšuje i obtížnost jednotlivých úkolů. Účastník nebo tým s nejvyšším počtem bodů vyhrává. Cílem je nalezení vlajky (flag) a zapsání výsledku do příslušného políčka a následné získání bodů. Vlajka se většinou nachází v souboru, který je potřeba nalézt. Po nalezení souboru, vlajku lze získat pomocí dešifrování nebo jiného zpracování souboru. Další způsob k získání vlajky je přistoupení k serveru určeného speciálně pro CTF, například pomocí SQLinjection. [3]
- Druhým typem je útok-Obrana, neboli **Attack-Defend**, kde se útočící tým pokouší vniknout do chráněného systému a získat vlajku, jako jsou textové soubory nebo obrázky. Tým pro obranu má za úkol zabránit k získání vlajek. Pro útočení jsou povoleny různé hackerské nástroje v rámci pravidel. Obranný tým má povoleno dělat cokoli pro obranu svých strojů, samozřejmě v souladu s pravidly. Není povoleno vypnutí rozhraní připojených do sítě nebo vypnutí samotných strojů. V rámci soutěže jsou obvykle uspořádány dvě kola, kde si týmy navzájem vymění role. [3]

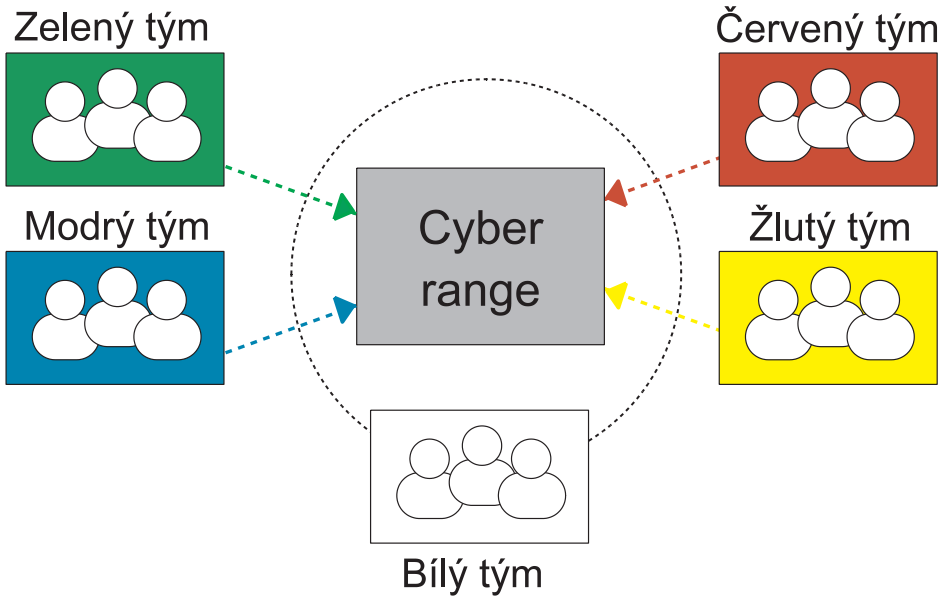
- Třetí typ, **King of the hill**, je zaměřen na zlepšení schopností v penetračním testování. Poskytuje situaci z reálného světa, kde profesionálové čelí kompromisům jako například ponechání běhu zranitelné služby nebo její vypnutí (například otevřený DNS resolver); získání přístupu k různým zařízením v síti. Hra se odehrává ve velkém izolovaném virtuálním prostředí skládající se ze zranitelných virtuálních strojů s operačními systémy Linux a Windows. Stroje jsou různě rozloženy a částečně propojeny pomocí podsítí [4]. V King of the hill jsou hráči rozděleni do jednotlivých týmů a je jim přidělena vlastní část infrastruktury [4]. Cílem je ochrana přiděleného území a získání území jiného týmu. K obraně vlastní infrastruktury lze používat: lepší zabezpečení vlastní infrastruktury nebo protiútoky na jiné týmy pro znemožnění útoku z jejich strany. Od typu Attack-Defend se liší v rozdělení rolí účastníků, kde skupiny nejsou pevně rozděleny na útočníky a ochránce. U King of the hill, je cílem ochrana své infrastruktury a získání nadvlády nad stroji jiného týmu, a to jakýmkoli způsobem dovoleným v pravidlech hry. [5]

1.2 Rozdělení účastníků

Během soutěže jsou účastníci rozděleny do týmů dle funkcí [6], viz Obr. 1.1:

- Účastníci odpovědní za běh infrastruktury patří do **zeleného týmu**. Dále mají na starost nastavení virtuálních počítačů a virtuální sítě. Monitorují dění a spravují případné poruchy.
- Organizátoři, rozhodčí patří do **bílého týmu** a řídí celkovou soutěž. Určují pravidla a přidělují úkoly. Mohou poskytovat i rady modrému týmu v případě potřeby.
- Skupina patřící do **žlutého týmu** představuje nezúčastněné uživatele generující nebezpečný obsah, například přistupování na nebezpečné stránky, stahování škodlivého softwaru.
- Skupina chránící zranitelnou infrastrukturu patří do **modrého týmu**. Během soutěže musí dodržovat určená pravidla. Většinou se jedná o studenty nebo zaměstnance, kteří mají za cíl zdokonalit své znalosti.
- Roli útočníků zastřešují profesionálové v **červeném týmu**. Dle předdefinovaného scénáře útočí na zranitelnosti. Za úspěšně provedený útok je modrému týmu je odebráno ze skóre.

Aktivně se soutěže zúčastní pouze modrý a červený tým, kteří si postupně zlepšují skóre. [7, 8]



Obr. 1.1: Rozložení týmů v cyber range

2 Technologie využívané v tréningových platformách

Tréningové platformy pro svůj běh využívají simulaci. Z modelů napodobující skutečné zařízení lze vytvořit infrastrukturu sloužící pro trénování. Pro tyto účely se používá technologie virtualizace, která umožňuje vytváření informačních služeb pomocí zdrojů tradičně vázaných na HW [9, 10]. Jsou používány dva typy: virtualizace na úrovni hardwaru nebo operačního systému-kontejnerizace. Za účelem simulace rozsáhlejší infrastruktury a jednoduššímu přístupu k platformě může být využit cloud computing.

Výše zmíněné technologie virtualizace na úrovni hardwaru, na úrovni operačního systému a cloud computing budou popsány dále společně s vybranými nástroji pro používání.

2.1 Virtualizace na úrovni hardwaru

Virtualizace na úrovni hardwaru vytváří simulaci fyzického prostředí a zdrojů počítače, jako je HW, datové úložiště, OS a počítačová síť. Tímto způsobem lze na jednom fyzickém počítači vytvořit více virtuálních strojů běžících nezávisle na sobě s vlastními virtuálními výpočetními zdroji a vlastním OS, viz Obr. 2.1 [11].

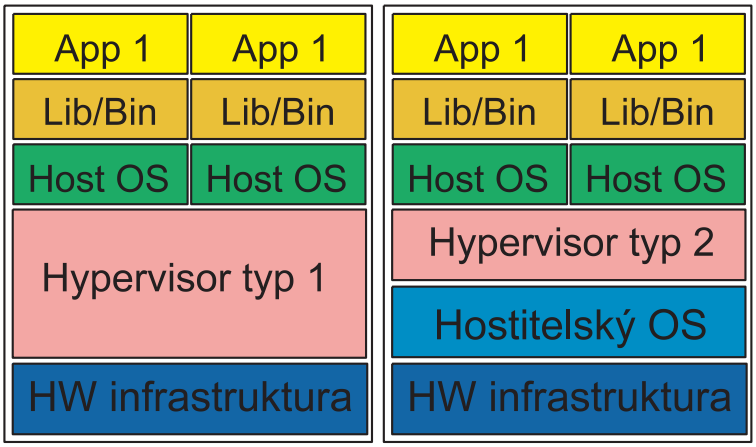
Výhodou virtuálních strojů je snížení nákladů na provoz fyzických zařízení a zvýšení efektivity využití zařízení, například plné využití výpočetních zdrojů [12]. Dále na jednom fyzickém počítači s daným operačním systémem lze současně spustit více virtuálních strojů s různým typem operačního systému nebo jinou architekturou. Virtualizace pomocí hypervizoru zajišťuje izolaci operačního systému jak od hostitelského, tak od jiného VM. Výhodu izolace lze využít při testování nebo spouštění programů z nebezpečných nebo nedůvěryhodných zdrojů nebo při testování aplikací na jiné platformě. Další výhodou je možnost přenosu VM na jinou infrastrukturu. Nevýhodou je nutnost virtualizace HW zdrojů spolu s OS a větší velikost přenášeného souboru na jinou infrastrukturu. [13, 14]

Vytvoření virtuálního počítače s operačním systémem a spolu s hardwarovými zdroji, se označuje jako hardwarová virtualizace [16]. Virtuální stroj je označován jako host (anglicky *guest*) využívající fyzických prostředků hostitele, fyzického počítače (anglicky *host*). Spuštěné aplikace na virtuálních strojích jsou odděleny od základních HW prostředků.

Pro vytváření VM na hostiteli slouží SW, nazývaný hypervizor, který zajišťuje správný běh VM, má na starost správu a izolaci od jiného VM, řízení a přidělování HW prostředků hostitele pro VM.

Existují dva typy hypervizorů: **typ 1** je hypervizor, který instaluje VM přímo na HW bez mezivrstvy s OS; **typ 2** je hypervizor, který vytváří VM nad vrstvou daného OS [17, 20].

Nejpoužívanějšími hypervizory jsou například VMware Workstation Player, VMware Workstation Pro, VirtualBox, Parallels Desktop, QEMU, XEN a Hyper-V [15].



Obr. 2.1: Struktura virtualizace-hypervizor typ1 a typ2

VMware Workstation Player a VMware Workstation Pro

VM ware Workstation Player a VMware Workstation Pro jsou vyvíjeny společností VMware Inc. Virtualizační nástroje jsou primárně určeny pro stolní počítače. V rámci typu hypervizoru patří pod typ 2. Umožňují vytváření VM na hostitelích s OS Windows a Linux. Například podporují Windows 10, Windows7, Ubuntu nebo CentOS. [24]

VMware Workstation Player poskytuje základní služby pro VM, například jejich zapnutí nebo základní nastavení sítě. Pro OS Linux v rámci služeb neumožňuje spouštět několik virtuálních strojů najednou. Dále Workstation Player nekoexistuje s VMware Workstaion Pro nebo VMwareServer na stejném hostitelském počítači. Tato varianta je poskytována bezplatně pro základní užívání. [24]

Naopak **VMware Workstation Pro** je placená a nabízí rozšířené nastavení VM. Patří mezi ně vytváření VM, vytváření virtuální sítě a její rozšířené nastavení, vytváření Snapshotů, klonování a sdílení VM přes vSphere, který je součástí VMware Infrastructure. Zkušební verzi si lze stáhnout na 30 dní. [24]

Obě varianty požadují minimálně 64bitový procesor s 2GB paměti RAM. V případě běhu více VM na jednom hostiteli je potřeba mít 4GB paměť RAM. VMware Workstation Player vyžaduje úložiště minimálně 150MB a Workstation Pro vyžaduje úložiště alespoň 1,2GB. [24]

Nastavení přídavných funkcí zajišťuje balíček **VMware Tools**. Je dostupný pro VM s OS Windows, MacOS, Linux a Solaris. Obsahuje nástroje a ovladače periferií, správu uživatelských procesů a vylepšení grafického výkonu virtuálních strojů. Stejně jako VirtualBox nabízí propojení hostitelského a hostujícího OS. Patří mezi ně například volný pohyb kurzoru myši mezi hostitelem a hostem, sdílení souborů, synchronizace systémového času a používání schránky.[25]
VMware Workstation Pro lze využít v rámci vytváření virtuálních strojů pro bezpečnostní cvičení, kvůli širší nabídce konfiguračních možností a kooperací
s VMware Infrastructure, která umožňuje virtualizaci rozsáhlé infrastruktury.
VMware Workstation Player je vhodnější pro spouštění již vytvořených a nastavených virtuálních strojů v rámci bezpečnostních cvičení. Virtuální stroje od VMware po nastavení VMWare Integration Pluginu lze také spouštět pomocí konfiguračního nástroje Vagrant. Pro jejich vzdálenou konfiguraci lze využít nástroj Ansible.

VirtualBox

Mezi další známé hypervizroy patří volně dostupný VirtualBox, vyvíjený firmou Oracle [27]. VirtualBox patří mezi typ 2 [17]. Je nabízen pro servery i pro stolní počítače [28].

Vyžaduje virtualizační technologie Intel VT-x a AMD-V a podporuje hostitele jako OS Windows, Linux, Solaris a MacOSX. Dokáže virtualizovat širokou škálu OS, například Windows 7, Windows Server 2003, Linux, Solaris a další.[28]

Aktivně je zveřejňován seznam funkcí a podporovaných hostitelských OS. VirtualBox je nabízen ve více jazycích. Umožňuje běh více virtuálních strojů na jednom OS najednou. Dále umožňuje jejich nezávislé zastavení nebo spuštění. V případě nastavení, virtuální stroje dokážou komunikovat mezi sebou, jinak jsou od sebe izolovány. V rámci vytváření VM, VierualBox nabízí tři formáty pro uložení diskového obrazu. Dále lze použít obrazy od jiných hypervizorů (VMware). Ještě jsou virtualizovány síťová karta a grafická karta. Při manipulaci s VM VirtualBox umožňuje ukládání aktuálních obrazů VM, neboli vytváření snapshotů, které lze zpětně vyvolat v případě potřeby pro navrácení se do předešlého stavu. Uložené snapshoty obsahují aktuální stav OS a jeho konfiguraci. Dále je možná specifikace parametrů pro HW zdroje a přiřazení VM do vytvořené sítě. Umožňuje vzájemné sdílení složek mezi hostitelem a virtuálním strojem. Dále podporuje připojení USB zařízení z hostitelského OS do virtuálního. [27]

V rámci bezpečnostních cvičení má VirtualBox rozsáhlé uplatnění. Výhodou je rozsáhlá možnost specifikace parametrů pro VM a široká škála podporovaných OS, jak pro hostitele, tak pro virtuální stroje. Pro vzdálenou konfiguraci spolupracuje s konfiguračními nástroji jako Vagrant a Ansible.

Parallels Desktop

Společnost Parallels vyvíjí řešení v oblasti virtualizace a automatizace [29]. Nabízí placený virtualizační nástroj typu 2 pro OS Mac Parallels Desktop. Od verze Parallels Desktop 11 je SW kompatibilní s Windows 10, Windows 8, Windows 8.1 a OS X El Capitan [30, 17].

Parallels Desktop nabízí virtualizaci OS Microsoft Windows, Linux nebo Google Chrome pro Mac. Nástroj nabízí tři režimy: Coherence, kde aplikace hosta a hostitele běží vedle sebe; Full Screen režim, kde je vytvořen plnohodnotný VM a Modality, který povoluje uživateli změnu velikosti okna VM.

V rámci režimu Coherence, pak MacOS nabízí ve svých programech i programy používané u hosta nebo rychlou možnost přístupu k hostu. Dále je umožněn přenos souborů a používání HW zařízení, díky propojení mezi systémy. [30, 31]

V rámci standardní edice je pro vytvoření VM povolen maximálně čtyř-jádrový procesor a 8GB RAM. Pro rozšířenou verzi je maximum třiceti-dvou-jádrový procesor a 128 GB RAM.[32]

Parallels Desktop je spíše určen pro možnost používání aplikací, které nejsou nabízeny nebo nejsou spustitelné na MacOS.

QEMU

Dalším virtualizačním nástrojem je volně dostupný QEMU, Quick Emulator, poskytující HW a SW virtualizaci. QEMU podporuje OS Linux, MacOS a Windows jako hostitelský systém. Současně virtualizace je podporována při použití KVM kernel modulu Linuxem nebo instalovaného hypervizoru XEN [35].

V případě, když QEMU je využit jako emulátor, tak dokáže emulovat OS využívající jinou architekturu pro testovací účely [33] co je možné pouze u OS Linux. Dále umožňuje běh programů nezávisle na architektuře pro jiný cílový OS [34].

Pro nastavení vytvářeného VM nabízí QEMU obdobné nastavení jako VirtualBox. Propojení přes síťový most umožňuje vytváření virtuální sítě mezi virtuálními stroji. Dále je umožněno propojení se síťovou kartou nebo USB portem. Pro vzdálenou konfiguraci také spolupracuje s konfiguračními nástroji jako Vagrant a Ansible. [36]

Výhodou je možnost konfigurace VM i přes terminál.

XEN hypervizor

Dalším využívaným hypervizorem je XEN, který patří mezi hypervizory typu 1 [17]. Původně byl vyvíjen univerzitou v Cambridge jako výzkumný projekt. Dnes je vyvíjen společností Linux Foundation ve spolupráci se společností Intel. XEN hypervizor je poskytován bezplatně pod licencí GNU GPLv2. [18]

XEN instaluje virtuální stroje do domén, kde *dom0* je první instalovaný VM, který dokáže přímo přistupovat k hardwaru. Většinou se jedná o OS s upraveným jádrem pro použití XENem. Další domény jsou pojmenovány jako uživatelské domény s označením *domU*, kde za *U* je dosazeno číslo. V rámci těchto domén běží další zvolené operační systémy. Z domény *dom0* lze konfigurovat další VM v doméně *domU*. [19, 21]

Hypervizor má na starost řízení a plánování běhu procesoru pro VM a v rámci svého použití podporuje migraci VM na jiného hostitele bez omezení. Virtuální stroje lze použít jako sandboxy izolované od jiných VM v oblasti bezpečnosti, například k analýze virů nebo červů. Virtualizační nástroj je také využíván i v cloudech pro poskytování privátních serverů. [19, 20]

XEN se hodí v rámci bezpečnosti na analýzu aplikací, programů nebo kódu v izolovaném prostředí. V rámci cloudů na VM můžou běžet platformy poskytující bezpečnostní cvičení. Pro vzdálené nastavení také spolupracuje s konfiguračními nástroji Vagrant a Ansible.

Hyper-V

Hypervizor Hyper-V od společnosti Microsoft také patří mezi hypervizory typu 1 [17]. Hyper-V umožňuje běh VM na hostiteli OS Windows s architekturou x86-64 [20].

Počínaje od Windows 8 Hyper-V nahradil předešlý hypervizor Windows Virtual PC. Jedná se o placený hypervizor. Microsoft nabízí i neplacené verze, které ale používají pouze terminál [17].

V první řadě, podobně jak u hypervizoru XEN, jsou virtuální stroje od sebe izolovány do sekcí a jsou závislé na prvně instalovaném virtuálním stroji jakou je u XENu *dom0*. Tady je *dom0* pojmenována jako rodičovská sekce-**parent partition**. V sekci **parent partition** běží Windows Server, který má přímý přístup k HW. Další virtuální stroje běží v sekci potomka-**child partition**. VM v sekci potomka má virtuální pohled na fyzické zdroje, ale nemůže k nim přistupovat. Může odeslat dotaz ohledně přístupu k nim do rodičovské sekce, která pak realizuje instrukce. Komunikace mezi jednotlivými sekcemi probíhá přes komunikační kanál VMBus. Rodičovská sekce disponuje se službou virtuálního poskytovatele, který komunikuje se službou virtuálního konzumenta v sekci potomka. [20] V rámci hypervizoru jsou poskytovány služby WMI Provider, která je zodpovědná za řízení a správu virtuálních strojů, VM Services a VM Worker Process. Worker Process poskytuje služby pro správu potomků ze strany rodičovské sekce. Dále pomocí VM Management Service jsou vytvořeny samostatné procesy pro každý VM. [20]

Hyper-V je poskytován pouze pro hostitele s OS Windows Server 2008 nebo vyšší ve verzích, Standard, Enterprise, Datacenter; Windows 7 nebo vyšší. Hardwarové

požadavky pro hostitelský systém jsou: architektura x86-64, možnost virtualizace pouze Intel VT-x nebo AMD-V. Další požadavky záleží od konkrétní verze OS. [22] Jako hosty Hyper-V podporuje: 10 vydání serverů od OS Windows (například Windows Server 2012, Windows Server 2008), 4 verze OS Windows pro stolní počítače (například Windows 10 nebo Windows 7), dále jsou podporovány OS Linux. [22]

Hypervizor vytváří izolované VM a tím umožňuje vývoj a testování aplikací nebo OS. Nabízí snadnou konfiguraci a efektivní využití zdrojů. Hyper-V umožňuje přesun VM na jiný server bez přerušení a snadné zálohování. Dále pomocí virtuálních přepínačů je umožněna komunikace virtuálních strojů mezi sebou. [23] Dále VM lze konfigurovat pomocí nástrojů Vagrant a Ansible.

Z pohledu bezpečnostních cvičení je Hyper-V možné využít v případě infrastruktury obsahující pouze stroje s OS Windows.

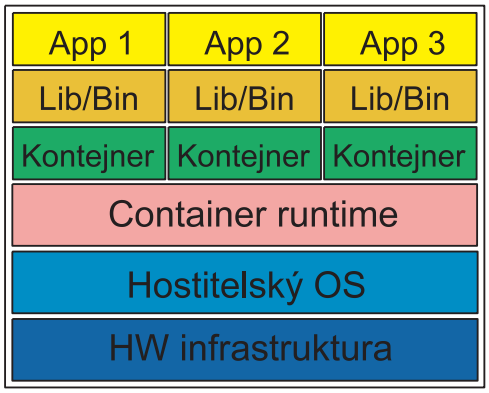
2.2 Virtualizace na úrovni operačního systému

Kontejnerizace je virtualizace na úrovni operačního systému. Místo hypervizoru se používá software Container runtime (například Docker) pro vytváření a správu kontejnerů. Virtualizují se pouze aplikace, knihovny bez HW zdrojů pro jednotlivé OS, které jsou pak rozděleny do jednotlivých kontejnerů sdílející hostitelský OS, její zdroje a knihovny. Spoluužíváním zdrojů se značně snižuje reprodukce kódu operačního systému a tím i využití místa na úložištích. Z důvodu sdílení hostitelského OS je zajištěn rychlejší start jednotlivých aplikací, tzn. nemusí se alokovat zdroje pro nový OS a následně spustit. Následující Obr. 2.2 zobrazuje rozložení bloků použitých pro kontejnerizaci. [11]

Nevýhodou kontejnerů je, že musí být navrženy pro stejný OS, na kterém běží. V opačném případě je nutné použít jiného hostitele. Další nevýhodou je možná zranitelnost jádra, které je využíváno všemi kontejnery. [13, 14]

K nasazení kontejneru je potřeba vytvořit *image* kontejneru, oddělit systémové zdroje pro souborový systém a konfigurovat síťová rozhraní. Dalším krokem je správa přidělování jádra a paměti pro jednotlivé kontejnery. [12, 37]

Využívanými open-source softwary pro kontejnerizaci jsou Docker a Linux Containers (LXC), které budou popsány dále.



Obr. 2.2: Struktura kontejnerizace

Docker

Docker je open-source SW a je určen pro vytváření, vývoj a řízení kontejnerů na běžném OS. Dále umožňuje zabalení všech příslušných balíčků, konfiguračních souborů a dalších součástí do jednoho kontejneru. [39]

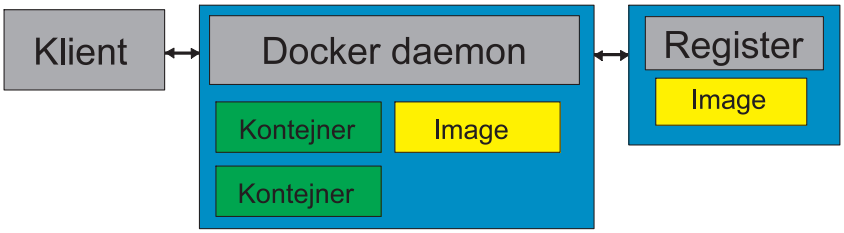
Již vytvořený kontejner sdílí služby jednoho operačního systému, na kterém běží. Tím, že kontejner obsahuje pouze aplikační soubory bez operačního systému je snížena velikost, režie pro běh a manipulace s kontejnerem. [39] Velkou výhodou Docker je rychlost vytvoření kontejneru, přenositelnost kontejneru beze změny. Další výhodou je možnost běhu více kontejnerů než virtuálních strojů na jednom hostiteli. Nevýhodou je svázanost s Linuxem jako hostitelským OS. Kontejnery nelze nasadit na 32-bitové OS pouze na 64-bitové. Ze strany Dockeru je nutné poskytnutí virtualizovaného prostředí pro OS Windows a MacOS. [39]

Docker pracuje na principu klient-server, viz Obr. 2.3. Klient komunikuje s Docker démonem, který zpracovává objekty, spouští a spravuje kontejnery. Klient je využíván jako prostředník mezi uživatelem a celým Docekrem. Pro komunikaci mezi nimi jsou používány příkazy. [38]

Jako úložiště obrazů aplikací, *image*, je využit *register*. *Image* slouží jako šablona pro vytváření kontejnerů. K sestavení aplikace se využívají objekty, mezi které patří například kontejner, který umožňuje spouštění aplikací. [40]

Docker byl primárně podporován operačním systémem Linux, ale z důvodu zájmu se jeho nabídka rozšířila i pro jiné operační systémy jako Microsoft Windows nebo Mac OS X [41].

Využití Dockeru se nabízí při instalaci některých tréningových platforem, kde je možné rozdělit služby do více kontejnerů.



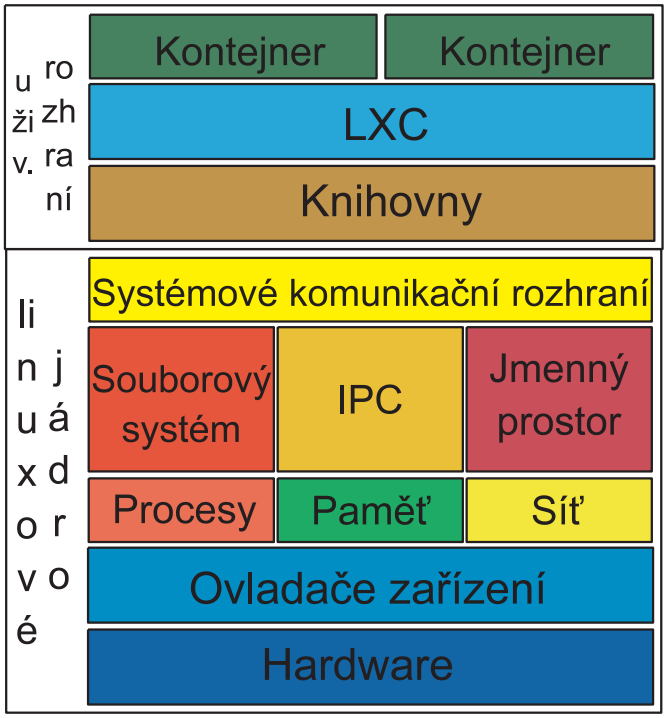
Obr. 2.3: Struktura Dockeru

Linux Containers

Linux Containers je open-source SW umožňující kontejnerizaci na OS Linux. Umožňuje běh několika izolovaných kontejnerů (prostředí) na jednom jádře. LXC zajišťuje vytvoření odděleného jmenného prostoru, síťových zařízení pro každý kontejner zvlášť, dále nastavuje IP adresu jednotlivě pro každý kontejner. [42]

Linux Containers sestává z několika komponent: knihovna *liblxc*, jazykové komponenty pro běh, jako jsou python3 nebo jazyk ruby; nástroje pro správu kontejnerů a šablony [44].

Následující obrázek Obr. 2.4 zobrazuje strukturu LXC. Ve spodní části je sdílený HW a procesy kontejnerů spravované jádrem systému. Ve vrchní části se nachází samotné LXC a vytvořené kontejnery, které může uživatel konfigurovat a spouštět [43].



Obr. 2.4: Struktura LXC

Pro nasazení kontejneru je prvním krokem instalace LXC, pak následuje nastavení síťového mostu a přidání cesty pro směrování. Následuje kontrola požadavků pro kontejner. Pak se přistupuje k vytvoření samotného LXC kontejneru dle předlohy. Předloha obsahuje údaje potřebné pro vytvoření prostředí, které je specifikováno ve zdrojovém *image*. V dalším kroku se kontejner spustí. [43]

Výhody tohoto SW jsou hlavně v jeho uživatelské přívětivosti a ve stejném používání funkcí od předchůdců kontejnerových platforem [45]. Nevýhodou je v podpoře pouze OS Linux jako hostitelského systému a tím podpora pouze aplikací, které běží na OS Linuxu.

2.3 Cloud computing

Komfortní způsob využívání sdílených výpočetních zdrojů přes internet nabízí model cloud computingu. Eliminuje náklady potřebné na koupi nového HW a SW. Zastřešuje kompletní správu, nastavení, zabezpečení a zálohu dat a tím zvyšuje produktivitu podniků, které jej využívají. K datům v cloudu lze přistupovat vzdáleně, přes webový prohlížeč, nebo přes jiného klienta. [47]

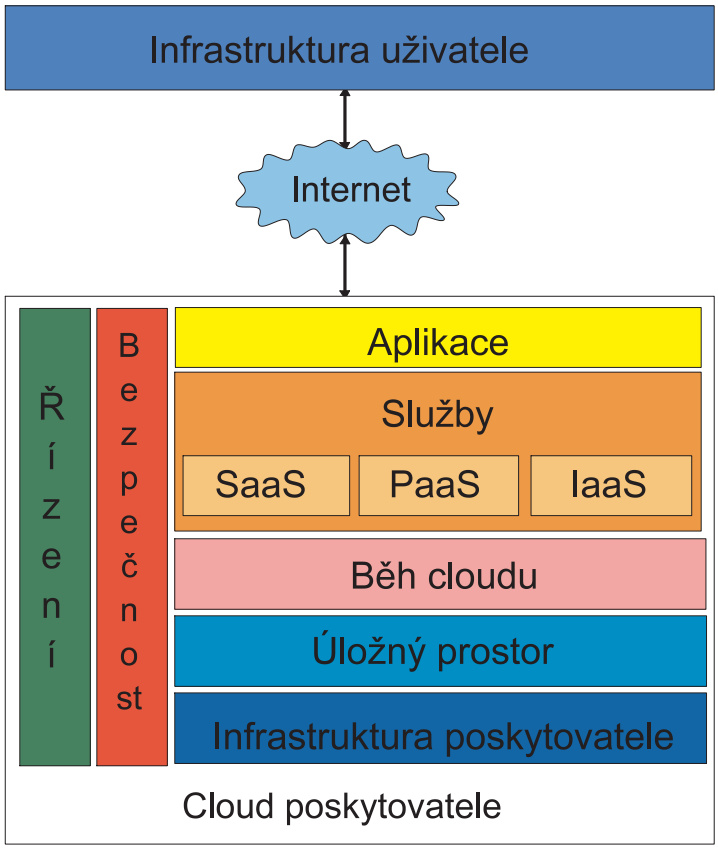
Virtualizace je technologie umožňující vytvářet více simulovaných prostředí na jednom HW systému. Oproti tomu cloud computing je soubor propojených a virtualizovaných HW zařízení (serverů), které dynamicky nastavovány a prezentovány jako jeden celek. Ve zkratce řečeno, cloud computing využívá technologii vitrualizace pro svůj běh. [48] V rámci aplikování cloud computing nabízí širokou škálu možností pro využití výpočetních zdrojů pro více uživatelů. Virtualizace zdrojů je určena pouze pro daný účel pro vybrané uživatele. K implementaci cloud computingu je využit SW jako OpenStack, ApacheCloudStack nebo OpenNebula [49].

Z pohledu nabízených služeb distribuční model cloud computingu popisuje: infrastrukturu jako službu (IaaS), platformu jako službu (PaaS) a SW jako službu(SaaS) [48]. Služby jsou popsány dále:

- **IaaS - Infrastructure as a Service**
  Poskytovatel poskytuje virtualizační platformu, kde je možné nasazení vlastního softwaru pro vytváření a řízení vlastních virtuálních strojů. Je nabízena fyzická infrastruktura, jako paměť nebo síťové zdroje. V rámci IaaS jsou pro řízení určeny softwary jako OpenNebula, CloudStack a OpenStack, které budou popsány dále v této kapitole [50].
- **Paas - Platform as a Service**
  Je poskytována platforma, která umožňuje nasazení aplikací a služeb zákazníkem. Lze využít nástroje, prostředí a programovací jazyky, které jsou podporovány dodavatelem. V rámci poskytované platformy je také poskytována infrastruktura, nad kterou ale zákazník nemá kontrolu.

- **Saas - Software as a Service**
  Nabízí se již vytvořené aplikace, které běží v cloudu. Většinou se jedná o e-mailové služby, ke kterým se lze připojit z jakéhokoli zařízení a místa. V rámci této služby zákazníci nespravují aplikaci nebo infrastrukturu.

Obr.2.5 ukazuje potřebné a využívané komponenty cloud computingu. Část *infrastruktura uživatele* slouží k přístupu ke cloudu přes zařízení na straně klienta. Může se jednat o počítač, tablet nebo server. Připojení do cloudu probíhá přes internet, většinou pomocí webové stránky-část *aplikace*. Jak bylo popsáno výše cloud nabízí služby, které jsou zvoleny ze strany uživatele-část *služba*. Služby a virtuální stroje jsou spouštěny a udržovány v části *běh cloudu*. Do *úložného prostoru* jsou uložena data využívaná v cloudu. Komponenta *řízení* se stará o řízení a koordinaci veškerých dalších komponent cloudu. Komponenta *bezpečnost* slouží pro nastavení bezpečnostních mechanizmů cloudu. Poslední komponentou je *infrastruktura poskytovatele*, která poskytuje HW a SW pro ostatní komponenty. V rámci HW se jedná o servery, úložné prostory nebo síťové prvky. SW zahrnuje technologie využívané v cloudu jako je například virtualizace. [51]



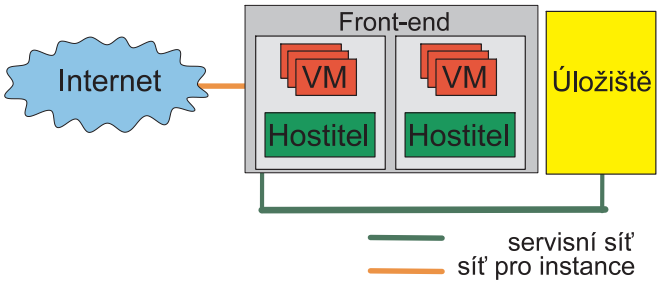
Obr. 2.5: Architektura pro cloud computing

OpenNebula

Volně dostupný software OpenNebula patří do skupiny poskytující *infrastrukturu jako službu*, kde poskytuje řízení a správu v rámci cloudu. Konkrétně se jedná o řízení virtuální infrastruktury v rámci privátních, veřejných a hybridních cloudů. Umožňuje běh virtuálních strojů a kontejnerů v rámci jedné platformy a pomocí libovolného virtualizačního nástroje. [54, 52]

K nasazení virtuálních strojů v rámci infrastruktury se využívají technologie, jako virtualizace, síťové technologie, bezpečnostní technologie nebo poskytování úložného prostoru. V rámci cloudové infrastruktury se nabízí možnost použití různých typů úložišť, hypervizory (například od společnosti VMware, XEN) a jiné HW a SW kombinace. [52]

Architektura OpenNebule sestává z několika komponent, které jsou vzájemně spojeny, viz Obr. 2.6. První komponenta se nazývá *fornt-end* a spouští všechny služby. Komponenta, která umožňuje řízení a sledování přes webové rozhraní se jmenuje *Sunstone*. Další komponentou je **hostitelský systém**, na kterém běží vybraný hypervizor. Tato komponenta spolupracuje s hypervizorem, umožňuje běh VM a řídí virtuální síť pro virtuální stroje. Výhodou je, že hostitelský systém může běžet na libovolné podporované platformě. *Hostitelský systém* musí mít připojení k *front-end* části. Více hostitelských systémů sdílející úložiště a síť jsou seskupeny do **klastru**. Komponenta **image repository** představuje úložiště pro obraz virtuálního stroje. Dále následují komponenty pro možnost správy a řízení pro různé technologie v rámci nasazení cloudu. Komunikaci mezi řídícími rozhraními umožňuje komponenta *OCA*. Komponenty modelu jsou propojeny sítěmi. K řízení a správě *hostitelských systémů* se používá *servisní síť*. Přístup k virtuálním strojům na *hostitelském systému* z internetu zajišťuje *síť pro instance*. [53]



Obr. 2.6: Model-OpenNebula

Díky své modularitě je OpenNebula využíván v různých odvětvích, například jako průmysl, výzkumní laboratoře, výpočetní centra nebo u poskytovatelů cloudů [52].

Apache CloudStack

Dalším volně dostupným software poskytovaným jako *infrastruktura jako služba* je Apache CloudStack, který je používán ve veřejných, privátních a hybridních cloudech. ApacheClouStack je určen pro řízení služeb v rámci cloudové infrastruktury (výpočetní řízení, řízení úložiště a síťového připojení) [55]. Ve většině je využíván poskytovateli veřejných cloudů nebo pro řešení hybridních cloudů. CloudStack poskytuje funkce: správa výpočetních zdrojů, správa uživatelů i účtů, otevřené rozhraní pro vytváření aplikací využívající data z OS nebo aplikace, síť jako službu a přívětivé uživatelské rozhraní.[57]

Architektura CloudStacku sestává ze serveru pro správu a řízení, Management server, a serveru se zdroji, které je potřeba řídit. Na druhém serveru se zdroji běží hypervizor pro virtualizaci zvoleného OS. Management Server pro správu a řízení přiděluje IP adresu pro VM, spravuje obrazy disků nebo snapshoty. [58]

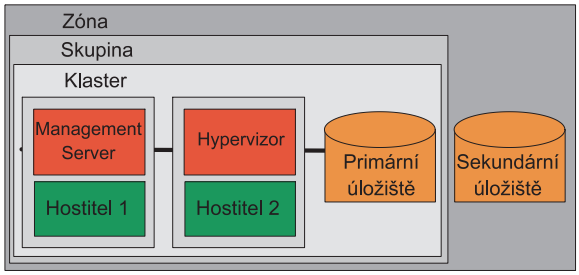
CloudStack pro instalaci vyžaduje minimálně dva servery: první stroj pro běh CloudStack Management Serveru a druhý pro nastavení cloudové infrastruktury pomocí hypervizoru [56]. Dále v rámci virtualizace podporuje dostupné hypervizory určeny pro běh serverů (například od společností VMware nebo Microsoft). Pro správu cloudu je umožněn přístup pro uživatele přes webový prohlížeč, příkazový řádek nebo vytvořeného rozhraní RESTful API.[57] CloudStack lze instalovat i na více uzlů, multinode, kde je pak počet hostitelů mnohem větší [58].

Celý CloudStack sestává z několika částí, viz Obr. 2.7 [58]:

- Hostitel - jeden výpočetní uzel v klastru, většinou se jedná o hypervizor,
- Primární úložiště - úložiště, které je poskytováno v rámci jednoho klastru,
- Klastr - obsahuje jedno nebo více hostitelů s primárním úložištěm,
- Skupina - obsahuje více klastrů propojených L2 přepínačem,
- Sekundární úložiště - poskytuje prostor pro ukládání obrazů disků nebo snapshotů,
- Zóna - sestává z několika skupin a sekundárního úložiště, většinou se jedná o oblast jednoho datacentra,
- Regiony - jedno nebo více geograficky blízkých zón, které jsou spravovány jedním nebo více Management Servery.

V rámci poskytování síťového připojení jsou možné dvě možnosti:

- Základní - hosté v rámci sítě jsou izolováni na L3 vrstvě přemostěním u hypervizoru,
- Rozšířené - hosté jsou již izolovány na vrstvě L2, například přes VLAN.



Obr. 2.7: Architektura CloudStacku

OpenStack

Třetí volně dostupným SW ze stejné skupiny je OpenStack, který obsahuje sadu nástrojů pro vytváření a řízení platforem cloud computingu pro privátní a veřejné cloudy. Byl vytvořen v roce 2010 společností Rackspace a organizací NASA. [59] Poslední nejnovější verze jsou Usuri, Train, Victorie a Wallaby [75].

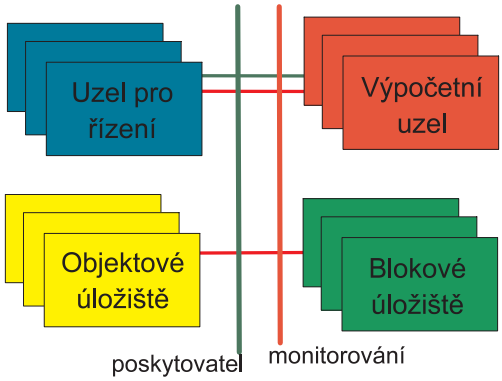
OpenStack nabízí využití virtualizace na hardwarové úrovni i na úrovni operačního systému-kontejnerizaci. Dle volby virtualizace je možno volit mezi několika instalačními nástroji, které budou popsány níže v části Architektura. OpenStack je tvořen uzly, pomocí kterých lze vytvářet instance pro různé účely. Uzly jsou rozděleny do rolí. V rámci rolí jsou uzlům přiděleny funkce, které následně poskytují jako službu. Role jsou rozděleny následovně: řídící, monitorovací, uzel pro síť a uzel pro ukládání, viz Obr. 2.8. Uzly pak zastřešují funkce:

- Uzel pro **řízení**, jinak kontrolér, slouží pro běh služby pracující s obrazy VM, řízení výpočetních uzlů, správu sítí, úložiště a řízení grafického uživatelského rozhraní.
- HW pro kontrolér je poskytován **výpočetními uzly**, které na základě požadavku vytvářejí kontejnery nebo virtuální stroje.
- Pro **monitorování** slouží síť managementu, který umožňuje komunikaci mezi všemi uzly.
- Síť umožňující připojení instancí k internetu se nazývá **síť poskytovatele**.
- Pro uložení souborového systému poskytovaného instancím je využito blokové **úložiště**. Objektové úložiště slouží pro ukládání účtů, objektů a kontejnerů. [12]

Pro fungování OpenStack vyžaduje alespoň dva uzly: řídící uzel a výpočetní uzel. Dále je možné přiřadit další přídavné uzly jako jsou například uzly pro úložiště.

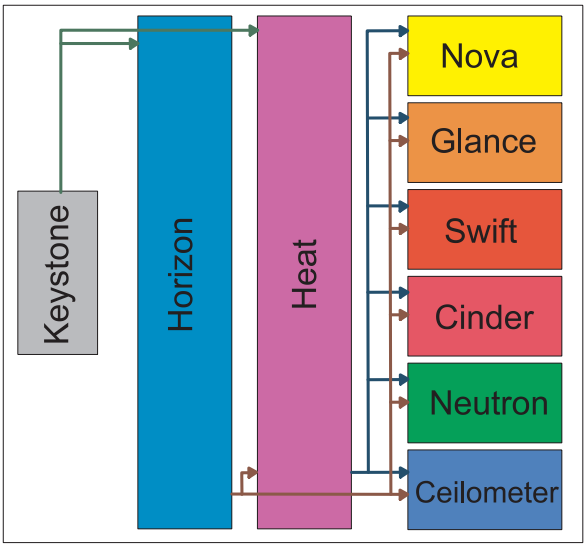
Architektura OpenStacku

Platforma OpenStacku sestává z několika komponent nabízející služby, které lze vzájemně kombinovat. Je tvořena ze součástí: Horizon, Keystone, Nova, Glance, Swift, Neutron, Cinder, Heat a Ceilometer, které jsou zobrazené na obrázku Obr. 2.9. [60]



Obr. 2.8: Role uzlů v OpenStacku

Službu pro autentizaci a povolení přístupu uživatelů provádí komponenta **Keystone** Jako uživatelské rozhraní je určena komponenta **Horizon**, která se stará o grafické zobrazení a nastavení aktuálně vybrané instance na pracovní ploše. Součást **Nova** je výpočetním uzlem a slouží pro vytváření a řízení jednotlivých VM. Komponenta **Glance** umožňuje registraci, spouštění instancí, které jsou uloženy jako *image* disku. O uložení jednotlivých souborů se stará oddíl **Swift**, jako objektové úložiště, a oddíl **Cinder** slouží jako blokové úložiště, které zajišťuje připojování a odpojování jednotlivých bloků k serverům. Komunikaci mezi službami zajišťuje komponenta **Neutron**. Pro orchestraci je určena součást **Heat**, tzn. zajišťuje automatickou konfiguraci, koordinaci a správu systému a softwaru [61]. **Ceilometer** má za úkol měření doby běhu jednotlivých instancí a zaznamenávání druhu použitých služeb, které jsou pak použity ve vyúčtování pro uživatele. Základními povinnými částmi pro používání jsou **Keystone, Nova, Glance, Swift a Neutron**. [59, 60]



Obr. 2.9: Architektura OpenStack

Instalace

OpenStack lze nasadit dvěma způsoby:

- **all-in-one** - celý instalační balíček bude instalovaný na jeden uzel, využívá se při testování,
- **multinode** - komponenty jsou instalovány na různé uzly.

Služby lze rozdělit do kontejnerů nebo nainstalovat přímo na uzly pomocí několika instalačních nástrojů dle konkrétních preferencí. Jedná se například o nástroje DevStack, TRIPLEO, OpenStack-Helm, Kolla-Ansible, OpenStack-Charms, Bifrost nebo další [62]. Dále v práci bude popsáno pět vybraných instalačních nástrojů, které se vzájemně liší jak od požadavků na HW, náročnosti instalace, tak s možností instalace přímo na HW nebo na VM.

DevStack

Prvním instalačním nástrojem je **DevStack**, který obsahuje soubor rozšiřitelných skriptů pro rychlou a kompletní instalaci OpenStacku, založenou na nejnovější verzi. Je určen pro účely nekontejnerizovaných systémů pro all-in-one nasazení. [63]

Pro instalaci OpenStacku přímo na OS je potřeba mít předinstalovaný OS Linux. DevStack se snaží podporovat různé distribuce Linuxu, jako je poslední verze Fedory nebo CentOS, ale doporučenou verzí je Ubuntu 18.04. Pro běh je požadována paměť RAM alespoň 4 GB, alespoň dvou-jádrový procesor a úložiště s kapacitou 10 GB. Dále je nutné mít připojení k internetu a uživatele s administrátorskými právy. Po splnění předchozích podmínek následuje přidání uživatele pro DevStack, stažení a konfigurace DevStacku. Po instalaci základních komponent pro OpenStack je možné se k SW připojit přes webové rozhraní. [65, 64]

Kolla Ansible

Druhou možností je **Kolla Ansible**. Podporuje nasazení all-in-one nebo multinode do Docker kontejnerů. Obrazy pro kontejnery se spravují pomocí nástroje **Kolla**. Pro nasazení kontejnerů jsou dostupné příručky. Požadavky na HW jsou 2 síťová rozhraní, 8 GB operační paměti a 40 GB úložiště. V rámci instalace jsou podporovány hostitelské operační systémy CentOS a Ubuntu. [69]

PackStack

Třetí možností je použití nástroje **Packstack**. Pomocí Packstacku lze OpenStack nasadit na jeden uzel nebo na více uzlů [70]. Instalaci lze provést přímo na OS nebo na VM s vyhovujícími parametry. Instalaci OpenStacku přes Packstack v současnosti lze realizovat pouze na OS CentOS a RHEL od verze 7, kde pro instalaci je podporována jen architektura x86-64. V rámci HW požadavků jsou potřebné tyto parametry: procesor s podporou virtualizace, paměť RAM v rozmezí 8-16GB, síťový adaptér a úložiště 100GB. [71, 72] Pro základní nastavení je potřeba pomocí terminálu instalovat verzi OpenStacku, vybrat počet uzlů a nastavit síť [71].

OpenStack-Helm

Čtvrtou možností je **OpenStack-Helm**, který umožňuje nasazení a údržbu volně spojených služeb OpenStacku. Tento nástroj podporuje OS Linux. V případě nasazení celého OpenStacku jsou HW požadavky následující: osmi-jádrový procesor, 16GB RAM, úložiště 48GB. V případě nasazení OpenStacku bez komponent Cinder a Horizon je potřebné mít čtyř-jádrový procesor, 8GB RAM a úložiště 48GB. Pro nasazení je nutné stažení balíčků, instalace a konfigurace SW pro řízení kontejnerů Kuberntes a správce balíčků Helm [66, 67]. Dále následuje nasazení jednotlivých komponent OpenStacku. Veškeré informace, vyjma již citovaných, byly čerpány ze zdroje [68].

OpenStack-Charms

Pátou možností je nasazení pomocí **OpenStack-Charms**, které vyžaduje instalaci SW MAAS a Juju. MAAS se používá pro nasazení cloudu na HW zdroje bez zásahu do OS. MAAS v rámci infrastruktury nasazuje:

- Uzly-jednotlivé stroje, které jsou řízeny řadičem,
- Rack-spravuje HW zdroje v rámci jedné skupiny,
- Region kontrolér-řídí celý region a spolupracuje s širším prostředím,
- Networking-slouží pro vytváření síťových topologií.

Následující tabulka Tab. 2.1 popisuje minimální HW požadavky na každou výše popsanou část:

Část infrastruktury	**RAM [GB]**	**CPU [počet]**	**Úložiště [GB]**	**Síť. rozhraní [počet]**
MAAS(Rack, Region)	8	2	40	1
Uzel s Juju	4	2	40	1
OpenStack 4 uzly	8	2	80	2

Tab. 2.1: OpenStack-Charms požadavky na uzly

Po instalaci MAAS se specifikují další parametry, jako nastavení hesla pro SSH připojení, DHCP server nebo přidání dalších uzlů. Po konfiguraci následuje nastavení uzlu pro Juju.

Program Juju slouží pro modelování základní infrastruktury pro OpenStack. Umožňuje nasazení, smazání a znovu nastavení infrastruktury. Po instalaci Juju jsou uzly MAAS přidány do Juju pro možnost správy. Po vytvoření Juju kontroléru se přistoupí k vytváření modelu pro OpenStack. V dalším kroku se přistoupí k instalaci OpenStacku, a to pomocí Juju, který nasadí a vzájemně propojí komponenty pro OpenStack.

Nabízí se dvě možnosti. První, jednodušší, která popisuje fungování celého mechanizmu. Druhá možnost je pro pokročilé, kde se OpenStack instaluje automatizovaně jako celek. V rámci Juju se vybírají komponenty a spojují k sobě. Dále

následuje konfigurace samotného OpenStacku. Po celkovém nastavení se lze přihlásit k OpenStacku přes webový prohlížeč.

Veškeré informace týkající se nasazení OpenStacku pomocí OpenStack-Charms byly čerpány ze zdroje [73].

Srovnání popsaných variant pro instalaci OpenStacku

V následujících tabulkách Tab. 2.2 a Tab. 2.3 jsou srovnány varianty instalačních nástrojů dle různých kritérií. Ze srovnání můžeme vyvodit, že OpenStack lze nejjednodušeji instalovat pomocí DevStacku a PackStacku, **náročnost instalace** hodnota 1, a nejsložitější na instalaci jsou varianty OpenStack-Helm a OpenStack-Charms, **náročnost instalace** hodnota 3. V rámci **náročnosti instalace** je zahrnuté množství provedených úkonů a odhadovaná doba nasazení. V rámci varianty nasazení DevStack nenabízí multinode nasazení vůči ostatním variantám.

V případě volby kontejnerizace lze použít varianty Kolla-Ansible nebo OpenStack-Helm. Výhodou DevStacku, PackStacku a OpenStack-Charms je možnost instalace přímo na HW bez OS. V případě volby instalace OpenStacku na virtuální stroj je možné zvolit DevStack nebo PackStack, díky tomu je možné používat OpenStack i pro testování. Nevýhodou variant pro instalaci je, že nepodporují OS Windows. Výběr varianty instalace závisí jen od účelu nasazení a dostupných HW zdrojů.

Varianta instalace	**Varianta nasazení**	**Typ nasazení**	**Náročnost instalace[1-3]**
DevStack	all-in-one	přímo na HW/VM	1
Kolla-Ansible	all-in-one/multinode	kontejnery Docker	2
PackStack	all-in-one/multinode	přímo na HW/VM	1
OpenStack-Helm	all-in-one/multinode	kontejnery Kubernetes	3
OpenStack-Charms	all-in-one/multinode	přímo na HW	3

Tab. 2.2: Srovnání popsaných variant pro instalaci OpenStacku dle nasazení

Varianta instalace	**Podporovaný typ OS**	**min.RAM [GB]**	**min.CPU [počet]**	**min.úložiště [GB]**
DevStack	Ubuntu 18.04/14.04	4/8	2	10/60
Kolla-Ansible	CentOS/Ubuntu	8	2	40
PackStack	CentOS/RHEL od v7	8/16	2	100
OpenStack-Helm	Linux	16/8	8/4	48
OpenStack-Charms	-	4/8	2	40

Tab. 2.3: Srovnání popsaných variant pro instalaci OpenStacku dle požadavků na HW

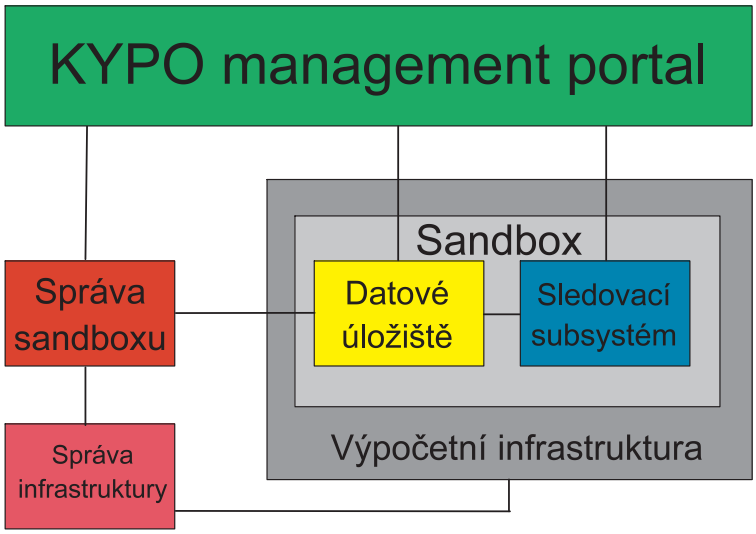
3 Tréningové platformy

V této kapitole budou představeny a popsány dva vybrané tréningové platformy využívající některou z výše zmíněných technologií.

KYPO

Platforma KYPO slouží k trénování a výzkumu kybernetických útoků a umožňuje simulaci počítačové infrastruktury pro zlepšení kybernetických experimentů [2]. KYPO je možné využít pro bezpečnostní cvičení, kybernetickou obranu, forenzní analýzu a síťovou simulaci [77]. Platforma KYPO je vyvíjena od roku 2013 Masarykouvou univerzitou ve spolupráci s NÚKIB, Ceritem, Ministerstvem vnitra ČR a projektem Concordia [79].

KYPO je navržena jako vícevrstvá aplikace, kde nejnižší vrstvu tvoří výpočetní infrastruktura, střední vrstvu tvoří sandboxy s datovým úložištěm a sledovacím subsystémem a vrchní vrstvu tvoří KYPO management portal, který umožňuje zobrazení a správu sandboxu. Vytvořené tréningové aktivity jsou monitorovány a ukládány pro příští využití. Přístup pro koncové uživatele k platformě je umožněn přes KYPO portál, kde lze provádět změny služeb daného úkolu tj. správa sandboxů, a lze přistupovat k jednotlivým sandboxům. Portál je uživatelsky přívětivý jak pro profesionály, tak pro studenty. Lze k němu přistupovat přes webový prohlížeč. Ve vrstvě sandbox jsou umístěny virtuální stroje propojené počítačovou sítí. Každý sandbox je izolován pro bezpečné provádění zadaných úkolů. Jednotlivé sandboxy jsou sledovány vlastním monitorovacím subsystémem a data jsou následně uložena do příslušného sandboxu pro příští použití, viz Obr. 3.1. [80]



Obr. 3.1: Architektura platformy KYPO

Základem KYPa je cloudová platforma OpenStack, která byla již popsána. Pro automatizaci konfigurování lze využít nástroje jako Ansible nebo Puppet. Hardwarové nároky na instalaci platformy nejsou zmíněny. Scénáře jsou vytvořeny v strukturovaném formátu typu YAML a slouží jako základní dokument pro popis topologie a vlastností sítě, použitého softwaru a přístupových oprávnění. K vyzkoušení je poskytnuto několik šablon jako například útok DDoS nebo phishing. [77]

EDUrange

EDURange je framework založený na cloudu pro vytváření interaktivních úkolů zaměřených na bezpečnost. V rámci svého využití je zaměřen na výuku a zlepšení dovedností v oblasti etického hackingu a bezpečnostní analýzy. Framework byl založen nezávislou vládní nadací U.S. National Science Foundation, která podporuje základní vědecký výzkum pro univerzity nebo jednotlivce. [77]

Platforma pro svůj běh využívá AWS cloud, ke kterému uživatelé můžou přistupovat pomocí webového prohlížeče. EDUrange podporuje virtuální stroje s operačním systémem Linux, které jsou přístupné přes SSH spojení. Tím, že platforma využívá AWS cloud, obrazy VM jsou uloženy ve formátu AMI. Scénáře pro řešení jsou sepsány ve formátu YAML. [77]

Pro instalaci vlastního EDURange serveru je dostupný podrobný vysvětlující postup a defaultní scénáře. Dále jsou dostupné manuály pro instruktora i pro studenty. Instruktor používá VM pro instruktora, který umožňuje spouštění scénáře a nastavení vyhodnocení.[78]

4 Příprava experimentálního prostředí

Tato kapitola se zabývá přípravou a popisem vybraného experimentálního prostředí pro vytváření sandboxů pomocí nástroje sandbox-creator, který je defaultně určen pro platformu KYPO, a tím splňuje podmínku kompatibility a využívání OpenStacku. V první řadě bude popsáno fungování již nasazené platformy KYPO v laboratoři a její kooperace s nástrojem sandbox-creator. Dále bude popsán samotný sandbox-creator a jeho instalace. V poslední části této kapitoly budou popsány používané konfigurační nástroje pro vytváření sandboxů v rámci praktické části této práce.

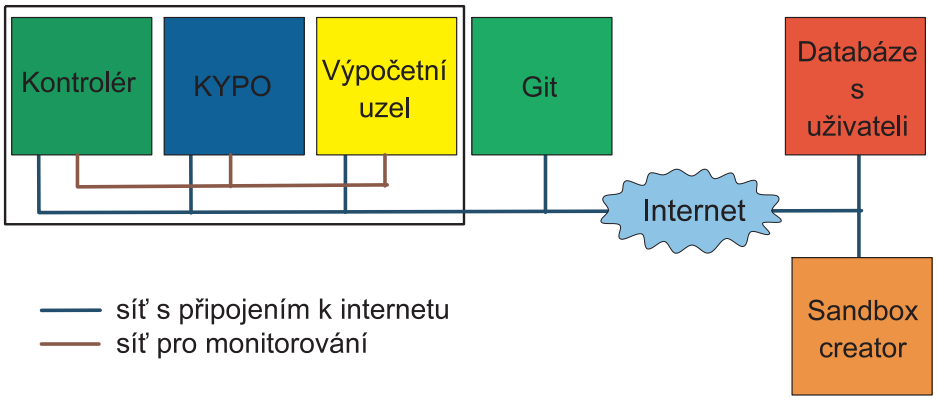
4.1 Sandbox-creator a platforma KYPO

Tato sekce popisuje již nasazenou platformu KYPO a její kompatibilitu se sandbox-creatorem popsaným v sekci Sandbox-creator struktura. Na uvedení popisu platformy a fungování její komponent byl udělen souhlas ze strany vedoucího práce pana Ing. Lieskovana.

Jak bylo v předchozí kapitole uvedeno, základem KYPa je OpenStack, který pro svůj běh potřebuje minimálně dva uzly: řídící a výpočetní. Pro zprovoznění platformy byly využity 3 stroje. Dva stroje pro OpenStack, kde na řídícím uzlu běží **kontrolér** k OpenStacku a na **výpočetním uzlu** běží Nova (Compute). Na třetím stroji běží **KYPO**, které má na starost nastavení tréningu a posílání instrukcí pro OpenStack. Celkový běh KYPa a součinnost jednotlivých komponent bude popsána níže. Dalšími potřebnými komponentami pro KYPo jsou:

- **databáze s uživateli**, kde je využit Open ID server, přes který jsou ověřováni uživatelé a jejich údaje a následně také ukládány pro přihlášení do KYPa,
- **Git**, na kterém jsou uloženy vytvořené sandboxy ze sandbox-creatoru.

Další částí jsou dvě sítě, pro rychlou interakci mezi stroji. Komponenty Kontrolér, KYPO a Výpočetní uzel jsou mezi sebou propojeny **sítí pro monitorování**, které mimo jiné slouží i pro jejich řízení. Dále každá komponenta je vybavena **připojením k internetu**. Poslední potřebnou částí je nezávislá komponenta na vytváření sandboxů pro KYPO, **sandbox-creator**, ze které jsou pak nahrány vytvořené konfigurační soubory na Git. Výše popsané části jsou zobrazeny na obrázku Obr. 4.1.



Obr. 4.1: Struktura již nasazené platformy KYPO

Dále následují kroky pro úspěšné načtení a zprovoznění sandboxu pro tréning na platformě KYPO:

1. Vytvoření sandboxu pomocí sandbox-creatoru na nezávislém stroji,
2. Nahrání a uložení sandboxu na Git,
3. Načtení sandboxu přes git na KYPO,
4. Oznámení ze strany KYPa kontroléru načtení a nastavení sandboxu,
5. Kontrolér nastaví na výpočetním uzlu základní parametry pro VM (OS, RAM, CPU, fyzická topologie sítě, atd.),
6. Kontrolér pošle výsledné hlášení o provedení úkolů na KYPO,
7. Instrukce ze strany KYPa pro Ansible pro provedení základní logické konfigurace sandboxu,
8. Pomocí Ansible se nastaví základní logické parametry (konfigurace sítě - nastavení IP adresy, masky, IP sítě),
9. Následuje User Ansible stage, kde probíhá dodatečná uživatelská konfigurace (instalace nástrojů, nahrávání souborů na VM, atd.),
10. Vytvoření tréningu přes KYPO portál,
11. Přiřazení sandboxů k tréningu,
12. Generování sandboxu v závislosti od počtu uživatelů.

4.2 Sandbox-creator struktura

Nástroj sandbox-creator se skládá z několika modulů, které zajišťují vytváření struktury pro sandbox, jako jsou generování konfiguračních souborů na základě požadované specifikace nebo vytváření složek pro jednotlivé definované zařízení. Další částí jsou šablony, ze kterých jsou vytvářeny konfigurační soubory pro daný typ zařízení. Do šablon je umístěno nastavení vytvořených hostů, jak dohromady, tak jednotlivě. Mezi šablonami se nachází i defaultní podoba konfiguračního souboru Vagrantfile

pro konfigurační nástroj Vagrant.

Dále v souborech typu YAML jsou uloženy předem nadefinované vlastnosti, které jsou využity během vytváření zařízení. Jedná se o soubor *flavors.yml*, který obsahuje nadefinované HW parametry ve formě různých kombinací velikostí RAM paměti a CPU. Například kombinace tiny1x2 má definovanou 2 GB RAM a jedno-jádrový procesor. *Interface.yml* obsahuje rozpis používaných síťových rozhraní pro konkrétní typ OS. Například síťové rozhraní pro VM kalilinux je eth1. Příkazy využívané pro definování vlastností v sandbox-creatoru jsou sdruženy s obdobnými příkazy využívané Vagrantem. Najdeme je v souboru *mapping.yml*.

Vstupním souborem pro vytvoření požadovaného sandboxu a popsaných virtuálních strojů v něm obsažených, je *sandbox.yml*. Obsahuje popis VM s jejich unikátním jménem, typem OS, HW parametry a přiřazením do sítě s příslušnou IP adresou.

Vstupním modulem, který spouští vytváření sandboxu je modul *create.py*. Celkové vytváření sandboxu je zajištěno ostatními moduly, které vychází ze specifikovaných parametrů ze souboru *sandbox.yml* a dalších příslušných šablon.

Specifikace ze *sandbox.yml* jsou mapovány na příkazy Vagrantu pro vytvoření Vagrantfile. Tento úkon je zajištěn modulem *attribute_formate.py*.

Modul *file_generator.py* zajišťuje načtení zadaných vlastností a vytvoření konfiguračních souborů, *main.yml*, pomocí šablon pro každé požadované zařízení. Dále je generován *playbook.yml*, který popisuje rozložení hostů v rámci vytvořených sítí. S využitím šablony je ze získaných dat ze souboru *sandbox.yml* vytvořen konfigurační soubor Vagrantfile. Konfigurační soubory, *main.yml*, jsou umístěny ve vytvořené složce *base_provisioning/jméno_hosta/tasks/* pro každé zařízení zvlášť.

Ve složce *base_provisioning/hosts/tasks/* je uložen soubor s nastavením, které se vztahuje na všechny hosty. *Device_creator.py* načítá nastavené vlastnosti pro zařízení a vytváří jejich seznam.

O konfiguraci sítí se stará modul *network_parser.py*. Provádí rozřazení hostů do příslušné sítě s danou IP adresou a nastavením síťového rozhraní. Nastavení směrování je vytvářeno pomocí modulu *routing.py*, kde je vytvořen a nakonfigurován hraniční router s parametry. Následně jsou mapovány cesty ze směrovačů na hraniční router.

O vytvoření VM pro VirtualBox se stará *provider.py*. Konfiguruje nastavené parametry pro hosty, jako HW specifikace. Dále je zodpovědný za nastavení obrazu OS routeru a dalších vlastností, jako jméno, velikost RAM a počet CPU.

Vytvořené sandboxy mají strukturu, která je požadována OpenStackem v rámci nastavování prostředí, tj. konfigurační soubory jsou rozděleny do složek dle typu konfigurace, které jsou pak načítány OpenStackem. Například, složku pro základní konfiguraci VM, složka pro nastavení sítě a složka pro dodatečnou konfiguraci.

4.3 Sandbox-creator instalace

Sandbox-creator umožňuje vytvoření oddělené infrastruktury zabalené do příslušného sandboxu. Vytvořený sandbox je pak dále možné nahrát na KYPO platformu, kde je obsah sandboxu s nastavenými virtuálními stroji zobrazen. Sandbox-creator pro svůj běh podporuje operační systémy Windows a linuxové distribuce založené na debianu. V našem případě byl zvolen OS Ubuntu 18.04. Pro běh sandbox-creatoru byla požadována instalace:

- programovacího jazyka **Python 3**,
- hypervisoru **VirtualBox 6.0**,
- konfiguračního nástroje **Ansible** aspoň verze 2.4,
- konfiguračního nástroje **Vagrant** verze 2.2.5

Na OS Ubuntu byl proveden update a upgrade dostupných balíčků, pomocí příkazů *sudo apt-get update* a *sudo apt-get upgrade*. Jazyk **Python3** byl instalován pomocí příkazu *sudo apt-get install python3*. Dále byl instalován **VirtualBox** z oficiálních stránek [90], kde před stažením nástroje bylo nutné importování veřejných klíčů. K tomu byly využity příkazy:

```
ubuntu@ubuntu:~# wget -q https://www.virtualbox.org/download/
oracle_vbox_2016.asc -O- | sudo apt-key add -
ubuntu@ubuntu:~#wget -q https://www.virtualbox.org/download/
oracle_vbox.asc -O- | sudo apt-key add -
```

Výpis 4.1: Importování veřejných klíčů

Dále byl přidán repozitář a nainstalované potřebné balíčky pro *linux-headers*, příkazem:

```
ubuntu@ubuntu:~# sudo apt-get install linux-headers-<verze>
-generic -y
```

Výpis 4.2: Přidání repozitáře pro linux headers

Následně bylo provedeno stažení samotného VirtualBoxu, příkazem *sudo apt install virtualbox-6.0 -y* [91]. Konfigurační nástroj **Ansible** byl instalován zadáním příkazu: *sudo apt-get install ansible*. Následující příkazy byly použity pro stažení příslušného balíčku s **Vagrantem** a instalaci Vagrantu [92].

```
ubuntu@ubuntu:~# curl -O https://releases.hashicorp.com/vagrant/
2.2.5/vagrant_2.2.5_x86_64.deb
ubuntu@ubuntu:~# sudo apt install ./vagrant_2.2.5_x86_64.deb
```

Výpis 4.3: Stažení a instalace Vagrantu

Dalším krokem byla instalace sandbox-creatoru. V případě stahování příslušných souborů sandbox-creatoru, bylo potřebné nainstalovat git: *sudo apt-get install git*.

Následně byla zadána cesta pro klonování projektu. V opačném případě bylo možno tento krok vynechat. Po stažení byl projekt otevřen v příslušné složce, *cd sandbox-creator*. Pomocí *sudo apt-get install python3-pip* byl nainstalován pip3. Následně byl zadán příkaz *pip3 install setuptools* a příkaz pro instalaci požadavků, *pip3 install -r requirements.txt*. Tímto byly nainstalovány všechny požadavky pro *sandbox-creator*.

4.4 Konfigurační nástroje

V této kapitole budou popsané konfigurační nástroje, které jsou využité v praktické části práce v souvislosti s vytvářením sandboxů.

Vagrant

Software, který slouží pro řízení a vývoj prostředí přes příkazový řádek za využití virtualizace [26, 81]. Vagrant je Vyvinut jako open-source software, společností HashiCorp [82].

Vagrant umožňuje konfigurovat více druhů OS přes příkazový řádek. Díky srozumitelné struktuře je nasaditelný na více druhů operačních systémů. Pracuje s virtuálními stroji s vlastním OS a knihovnami, které ale sdílí stejný HW.

Ze strany hypervisorů Vagrant spolupracuje s VirtualBoxem, VMwarem, AWS-EC2 a dalšími.

Vagrant pro běh VM využívá tzv. boxy, ve kterých se nachází daný OS. Boxy jsou multiplatformní za podmínky, že platformy jsou podporovány Vagrantem. Lze vytvořit vlastní box nebo lze stáhnout z rozsáhlého katalogu určeného pro Vagrant boxy. V případě stahování je nutné se odnavigovat přes terminál do kořenového adresáře projektu a následně zadat příkaz *vagrant init <jménoBoxu>* a spustit VM příkazem *vagrant up*. [83]

Příkaz *vagrant init <jménoBoxu>* vytvoří soubor, psaný v jazyce Ruby, zvaný Vagrantfile, ve kterém se nachází základní parametry pro daný box. V komentářích je napsané další možné nastavení, jako jsou velikost paměti, nastavení IP adresy a sítě. *Vagrant up* načítá tyto údaje a pomocí hypervizoru se nastaví parametry systému a provede se instalace. Po tomto úkonu virtuální stroj běží a lze s ním plnohodnotně pracovat. [84]

V případě dopředu nachystaných dodatečných nastavení se u příkazu *vagrant up* je spuštěn soubor pro provisioning. Dodatečné nastavení může zahrnovat vytvoření souboru na konkrétní místo na virtuálním počítači, nebo instalaci vybraného programu. Za podmínky, že soubor pro provisioning není defaultně přidán, lze provisioning nastavit až po námi vytvořeném souboru, pomocí příkazu *vagrant provision <jménoVM>*.

Pro připojení k VM přes Vagrant se používá příkaz *vagrant ssh*, které umožňuje bezpečné připojení přes SSH. Pro restart strojů Vagrant používá *vagrant reload*. Po nastavení veškerých náležitostí lze virtuální stroj vypnout pomocí příkazu *vagrant halt* nebo *vagrant halt <jménoVM>* v případě používání více strojů.

Ansible

Ansible je svobodný SW určený pro správu, konfiguraci a řízení cloudů, VM, aplikací a mnoho dalšího. Byl vyvinut Michaelem DeHaanem. [85]

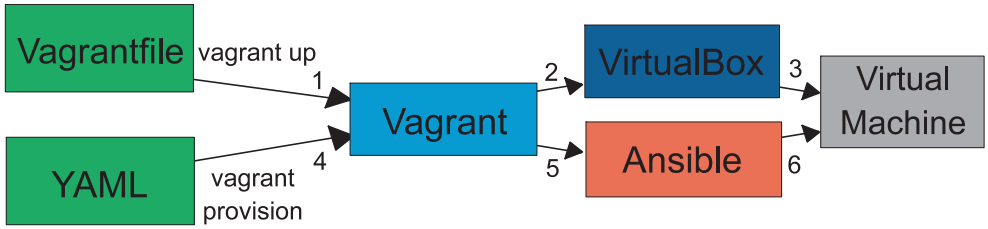
Ansible pro svůj běh používá dva typy serverů: řídící stroj a řízené uzly [87]. Uzly jsou synchronizovány pomocí modulů, které umožňují komunikaci s řídícím uzlem. Moduly umožňují opakované konfigurování a tím se docílí nastavení systému do stejného stavu. Konfigurace uzlů je popsána pomocí YAML souborů, nazvané také playbook. Pomocí těchto souborů lze uzlům nastavit požadované role. Automatickou konfiguraci na uzlech zajišťuje řídící stroj přes moduly, které jsou nahrané pomocí SSH spojení. [86]

Playbook obsahuje požadované parametry pro daný uzel. Může se jednat o načtení souboru na VM nebo instalace programu a jeho následná konfigurace. Zjednodušeně slouží jako sada instrukcí nebo úkolů pro daný uzel, který můžeme upravovat, vícekrát spouštět a uchovávat. Pro spuštění konkrétního playbooku se použije příkaz *ansible-playbook <jménoPlaybooku>.yml -f 10*. Následně se začnou postupně spouštět požadované úkoly. V průběhu vykonávání je vrácena hláška o stavu vykonané činnosti. Při neúspěchu je vykonávání úkonů pozastaveno pro daný uzel. Následně je provedena celková sumarizace v podobě hlášení. [88]

Před samotným spuštěním lze otestovat daný playbook pro eliminaci případných syntaktických chyb, příkazem *ansible-playbook –syntax-check <jmeno-playbooku>.yml* nebo *ansible-lint <jmeno-playbooku>.yml*. Pomocí příkazu *ansible-lint -L* lze vypsat číslování chyb dle typu.

Ansible lze instalovat na řídící stroje s operačním systémem jako je Debian nebo Ubuntu. Pro spravované uzly požaduje nainstalovaný jazyk Python: pro Linux verze 2.4 a vyšší. V rámci podpory cloud computingu dokáže pracovat na VM i v cloudech, například AWS nebo OpenStack. [87]

Následující Obr. 4.2 popisuje postup vytvoření VM pomocí načteného Vagrantfilu (bod *1*) Vagrantem. Bod *2* označuje instalaci VM a nastavení jeho parametrů, kde je výsledkem samotný VM (bod *3*). Bod *4* zahrnuje načtení dodatečného nastavení Vagrantem a předání Ansiblu (bod *5*), který uskuteční instrukce uložené v souboru YAML (bod *6*). [89]



Obr. 4.2: Využití konfiguračních nástrojů

5 Vytváření a zprovoznění sandboxů pro laboratorní cvičení

Po úspěšném zprovoznění experimentálního prostředí se přešlo k vytváření prostředí pro laboratorní úlohy. Po pečlivém uvážení byly vybrány témata: SSL-Strip a útoky DDoS. Byly vybrány z důvodu spolehlivé proveditelnosti v určitém časovém intervalu, tj. existují fungující postupy pro realizaci útoku pro tyto témata a tím nebylo nutné experimentování s realizací útoků a bylo soustředěno na vytváření samotného prostředí.

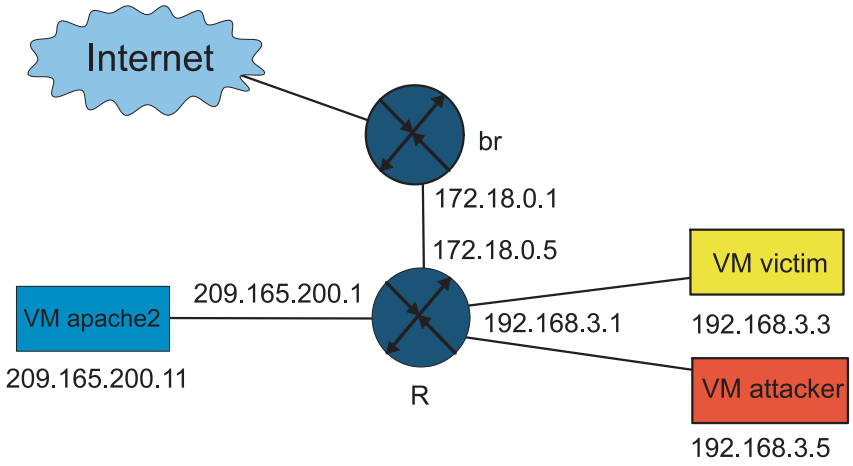
Na začátku popisování každého sandboxu bude krátce popsáno téma, pro které byl sandbox vytvořen. Dále bude následovat rozložení virtuálních strojů v síti, tzn. adresa sítě a VM. Pak bude popsáno vytvoření samotného prostředí pro jednotlivé virtuální stroje. Dále jsou v příloze této práce uvedeny vytvořené laboratorní úlohy pro tyto vytvořené sandboxy: B pro sandbox SSL-Strip a A pro sandbox DDoS.

5.1 Sandbox pro útok SSL-Strip

Realizace SSL-Stripu využívá slabinu protokolu SSL, který zajišťuje šifrovaný přenos od zdroje k cíli napříč počítačovou sítí. Cílem útočníka je dostat se mezi komunikující strany a získat jemu potřebné údaje, tzv. útok Man In the Middle. Většinou se jedná o získání přihlašovacích údajů z přihlašovacích formulářů na webových stránkách využívající aplikační protokol HTTP se zabezpečením pomocí SSL protokolu. [107, 112]

Na vytvoření sandboxu byly použity tři virtuální stroje, vyjma defaultně nastavených směrovačů, konkrétně: VM z vagrant boxu *ubuntu/xenial64* s příkazovým řádkem pro webový server s názvem **apache2**, VM *kalilinux/rolling* s grafickým rozhraním pro útočníka s názvem **attacker** a pro koncového uživatele s názvem **victim** byl použit box *milbuild/graphical-ubuntu16*, také s grafickým rozhraním. Jednotlivé stroje byly rozřazeny do sítí simulující reálnou situaci a byla jim přiřazena IP adresa. Spojovacím prvkem mezi sítěmi byl defaultně vytvořený a nakonfigurovaný router **R**. Přístup k síti internet zajišťoval také výchozí router **br**.

Webový server byl přiřazen do sítě využívající veřejnou IP adresu 209.165.100.0/24. Koncový uživatel a útočník byly přiřazeny do sítě s privátní IP adresou 192.168.3.0/24. Následně se nastavily konkrétní IP adresy pro jednotlivé stroje. Následující obrázek Obr. 5.1 zobrazuje síťovou topologii virtuálních strojů v sandboxu. Následuje tabulka Tab. 5.1 s detailnějším rozpisem pro virtuální stroje jako jsou: jména VM, vybraný OS, jejich rozložení ve vytvořené síti, přiřazenou IP adresu a přihlašovací údaje.



Obr. 5.1: Sandbox sslstrip - Síťová topologie virtuálních strojů

Virtuální stroj-OS	**síť**	**IP adresa**	**uživ. jméno-heslo**
apache2-ubuntu/xenial64	apache2-R	209.165.200.11	vagrant-vagrant
user-ubuntu16.04	victim-attacker-R	192.168.3.3	ubuntu-vagrant
attacker-kaliLinux	victim-attacker-R	192.168.3.5	vagrant-vagrant
R-debian	apache2-R	209.165.200.1	-
	victim-attacker-R	192.168.3.1	-

Tab. 5.1: Rozpis parametrů jednotlivých strojů

Výše uvedené údaje, vyjma přihlašovacích údajů, byly vloženy do souboru sandbox.yml, který slouží jako vstup pro vytvoření ostatních parametrů pro funkční sandbox. Následně pomocí příkazu *python3 create.py sandbox.yml* byl vytvořen Vagrantfile a ostatní soubory umožňující další konfiguraci pomocí Ansible, jak bylo popsáno v Struktura sandbox-creatoru. Dodatečná konfigurace pomocí Ansible byla provedena na virtuálních strojích **apache2, attacker, victim** a **R**.

Pro načtení playbooků pro jednotlivé VM bylo potřeba do hlavního playbooku *provisioning/playbook.yml* přidat několik řádků nařizující konfiguraci.

Konfigurace VM apache2

V dalším kroku bylo nutné nakonfigurovat webový server pro uskutečnění simulace. Ve složce *provisioning/roles/apache2/tasks* se vytvořil seznam požadovaných úkolů, které byly rozděleny do souboru dle vykonávané oblasti. Do hlavního playbooku *main.yml* byly vloženy odkazy na tyto soubory, které byly jednotlivě volány. Soubory se seřadily v navazujícím pořadí: *apache2.yml, python.yml, certificate.yml, set_web.yml, restart_apache2.yml*.

Modul apache2.yml

Soubor *apache2.yml* obsahuje základ pro úspěšné vytvoření prostředí. Jako první

se nainstaloval webový server *apache2*. Dále byl nastaven port v souboru */etc/apache2/ports.conf*, na kterém má *apache2* naslouchat. V našem případě je to port pro zabezpečený HTTP přenos s číslem 443, *Listen 443* a port pro nezabezpečený přenos, 80, z důvodu realizace přesměrování.

Dále byly nastaveny parametry pro zabezpečený přenos v souboru *ssl-params.conf*, pomocí nahrání před připraveného konfiguračního souboru na virtuální stroj. *Ssl-params.conf* obsahuje základní nastavení SSL protokolu. Jsou to například použití šifrování během přenosu SSLCipherSuite, které vychází z nainstalované sady OpenSSL; používané verze protokolu SSLProtocol a SSLHonorCipherOrder zajišťující dodržení šifrovacích předvoleb serveru [93].

Dále se nastavil virtuální host v konfiguračním souboru *default-ssl.conf* pro port 443. Bylo nastaveno jméno serveru *ServerName xsecret-page.com* a *ServerName 209.165.200.11*. *Default-ssl.conf* soubor ještě obsahuje výchozí cestu pro běh webové stránky */var/www/html*, nastavenou cestu ke klíči a k certifikátu. Uložení chybových hlášení, zapnutí ochrany pomocí protokolu SSL bylo ponecháno v původním stavu.

V konfiguračním souboru pro nezabezpečené připojení, *000-default.conf*, bylo nastaveno přesměrování na zabezpečenou stránku pomocí: *Redirect permanent "/" "https://<jméno webové stránky>/"* a *Redirect permanent "/""https://<IP adresa serveru>/"*.

Pro webovou stránku využívající jazyk PHP se musel nainstalovat hypertextový preprocesor PHP7. V souboru pro konfiguraci *apache2.conf* se nastavilo zpracování souborů s php koncovkou. Nakonec na firewallu byl povolen přenos přes port 443 a 80 pomocí TCP protokolu.

Modul python.yml

Modul *python.yml* slouží jako předpříprava pro následující modul *certificate.yml*. Obsahuje instalaci jazyku python a systém pro správu balíků pip na virtuálním stroji. Kvůli úspěšnému vytvoření certifikátu bylo nutné nainstalovat obě verze pipu, jak pip, tak pip3. Dále byl nainstalovaný balíček pro implementaci OpenSSL, *pyopenssl*, ve verzi 0.15 pro obě verze jazyka python. Následně byla provedena kontrola existence výše zmíněných balíčků. Jako poslední byl pip upgradeovaný na nejnovější verzi, kvůli požadavkům na vytvoření pyopenssl certifikátu v dalším modulu.

Modul certificate.yml

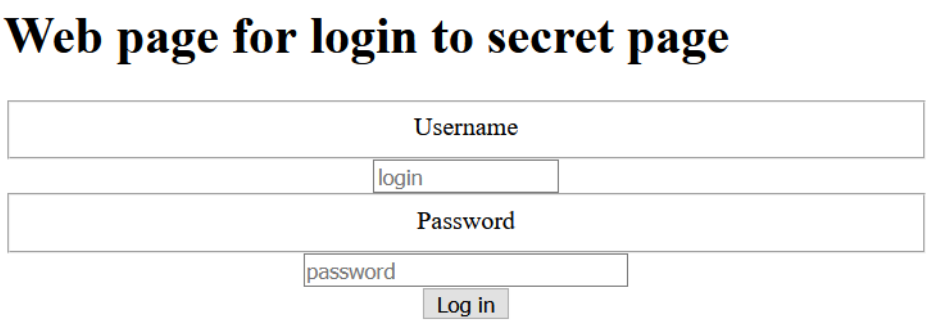
Modul *certificate.yml* obsahuje zprovoznění plně funkčního OpenSSL certifikátu. Je generován soukromý klíč, který je následně využit u generování certifikátu. Dále je vygenerován Certificate Signing request(CSR) pro certifikační autoritu, který obsahuje základní údaje o webové stránce a veřejný klíč. K vytvoření certifikátu byly využity výše zmíněné soubory.

Certifikát nebyl ověřen certifikační autoritou, ale v našem případě je dostačující. Pro distribuci certifikátu na klienta bylo nutné certifikát převést z formátu *.crt*

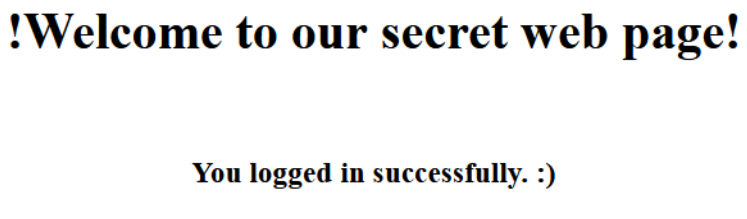
na formát *.pem*. Dále certifikát ve formátu *.pem* byl uložen do složky obsahující webovou stránku pro přístup uživatelů pro možnost jeho stažení.

Modul set_web.yml

Tento modul má za úkol nahraní vytvořených souborů na VM pro webovou stránku. Soubor *index.php* je titulní strana webové stránky, obsahuje přihlašovací formulář na "tajnou stránku". *Login_form.php* zpracuje požadavek a dle správnosti údajů je uživatel připuštěn na stránku. Jako poslední je soubor *secretPage.php*, který se objeví až po přihlášení. Na obrázku Obr. 5.2 je zobrazeno přihlašovací okno na vytvořenou webovou stránku. Další obrázek ukazuje vytvořenou stránku po přihlášení, Obr. 5.3



Obr. 5.2: Sandbox sslstrip - Webová stránka před přihlášením



Obr. 5.3: Sandbox sslstrip - Webová stránka po přihlášení

Modul restart_apache2.yml

Předposledním modulem pro nastavení virtuálního stroje **apache2** je *Restart_apache2.yml*. Modul obsahuje zapnutí nebo vypnutí nakonfigurovaných parametrů v souboru *apache2.yml*.

Pro korektní běh php bylo nutné vypnout modul *mpm_event* a zapnout *mpm_prefork* a *php7.0*. Zabezpečení pomocí protokolu SSL bylo zapnuté pomocí modulu *ssl*. Dále se zapnul modul *headers*, zapnul se virtuální host pro webovou stránku a parametry pro SSL připojení. Po těchto změnách byla provedena kontrola konfigurace pomocí

apache2ctl configtest. Po úspěšné kontrole byl *apache2* restartován. Posledním krokem tohoto modulu bylo zapnutí firewallu. **Konfigurace VM attacker**
Po úspěšné konfiguraci a vyzkoušení virtuálního stroje **apache2** bylo nutné nakonfigurovat VM **attacker**. Konfigurace byla provedena za účelem před připravení nástroje pro laboratorní cvičení. Instaloval se nástroj *bettercap*, který nabízí možnost realizace ARPspoofu i SSLstripu.

Konfigurace VM victim
Na stroji victim byl nainstalován balíček nástrojů *libnss3-tools*, obsahující nástroj *certutil* pro instalování certifikátů.

Konfigurace VM R
Dále bylo nutné vypnout přesměrování na směrovači R v souboru *base_provisioning/roles/routers/tasks/main.yml*, aby bylo možné provést útok ARPspoof. Do konfiguračního souboru */etc/sysctl.conf* byl přidán řádek pro vypnutí přesměrovávání, například při dotazu ping, *net.ipv4.conf.all.accept_redirect=0*.

Všeobecná konfigurace
Po nastavení webové stránky na serveru bylo nutné přidání nového aliasu v souboru na virtuálních strojích */etc/hosts*. Konkrétně přidání aliase pro webový server v playbooku pro *base_provisioning/roles/hosts/tasks*. Byl rozšířen řádek: *line: 209.165.200.11 apache2 www.xsecret-page.com xsecret-page.com*. Záznam byl pak následně nahrán na všechny stroje, aby dokázaly najít webovou stránku. Tento krok byl zvolen pouze z testovacích důvodů.

Nakonec byla provedena úprava i ve Vagrantfilu, kde bylo prohozeno pořadí virtuálních strojů z: **apache2, attacker, victim, br, R** na **br, R, apache2, attacker, victim**. Tohle prohození bylo provedeno, protože při spuštění celého sandboxu najednou, konfiguraci virtuálních strojů nebylo možné provést z důvodu nemožnosti připojení do sítě internet, tj. routery ještě neběžely a neumožňovaly přístup do sítě. Dále bylo u všech strojů vyjma směrovačů nastavena paměť na 1024MB, z důvodu omezené paměti na hostitelském počítači.

Následně byla vytvořena laboratorní úloha s popisem útoku a pro jeho obranu. Další částí úlohy je vytvořený a odzkoušený scénář pro červený tým a pro modrý tým pro realizaci vybraného útoku, na který byl sandbox vytvořen, viz příloha B.

Řešení potíží při testování
V průběhu vytváření a testování prostředí pro sandbox pro útok SSL-Strip, ale bylo nutné zdolat několik překážek.

V průběhu konfigurace certifikátu pro stanici **apache2** bylo nutné nainstalovat

obě verze pro pip a nainstalovat balíček *pyopenssl* taky pro obě verze jazyka python pro bezproblémové vytvoření certifikátu. Jak bylo výše popsáno na stanici **apache2** běží webový server poskytující stránku s přihlášením. Původní myšlenkou bylo uložení přihlašovacích údajů do MySQL databáze. Po úspěšném nastavení databáze a nahrání informací do ní na stanici **apache2**, se objevila překážka při přihlašování na stanici **victim**. Webová stránka byla zobrazena, ale po zadání přihlašovacích údajů se stanice **victim** nedokázala připojit na MySQL server. Z důvodu časově omezeného limitu pro vypracování práce bylo od tohoto záměru upuštěno a pro kontrolu přihlašovacích údajů bylo zvoleno umístění přihlašovacích údajů přímo do PHP souboru.

Po zprovoznění předešle zmíněných záležitostí bylo nutné vyzkoušet přesměrování z portu 80 na 443. Je nutné poznamenat, že pro načítání stránky se prvně používala IP adresa serveru. Pro přesměrování na HTTPS bylo nutné do prohlížeče zadat jméno webové stránky korespondující se jménem stránky v certifikátu jinak se stránka nenačetla z důvodů špatné autorizace. Počáteční myšlenkou bylo nastavení záznamu na DNS serveru, ale pro několik chyb a jejich časové náročnosti pro vyřešení, bylo zvoleno doplnění záznamu webové stránky do souboru */etc/hosts*. Tímto bylo úspěšně možné přesměrovat stránku z portu 80 na 443.

Dále u stanice **attacker** byl původní nápad pro realizaci útoku za pomoci nástrojů *arpspoof* a *sslstrip*. V případě použití nástroje *arpspoof* bylo přesměrování komunikace realizováno úspěšně. U nástroje *sslstrip* se vyskytly potíže, již při instalování potřebných balíků. Nástroj *sslstrip* je napsán v již méně používané verzi jazyka python, v python2, pro který bylo nutné doinstalovat pip2, pomocí kterého mohly být potřebné balíčky nainstalovány. Po úspěšné instalaci byla vyzkoušena realizace útoku a zachycení přihlašovacích údajů. Bohužel bez úspěchu. Za neúspěch bylo pokládáno špatné nastavení přesměrování na webovém serveru. Po kontrole a ujištění správné konfigurace, bylo přistoupeno ke změně nástroje používaného na stanici **attacker** pro realizaci útoku. Po vyzkoušení jiného nástroje se nástrojem pro přesměrování a zachycení přihlašovacích údajů stal *bettercap*.

Dále jak bylo napsáno v sekci Sandbox-creator instalace se byl požadován VirtualBox-6.0, který byl také nainstalován, ale v průběhu upgradu balíčků byla jeho verze zvýšena na 6.1. Z důvodu omezeného času VirtualBox nebyl instalován znovu ve verzi 6.0, ale byl ponechán ve verzi 6.1, se kterou se nevyskytly žádné potíže v průběhu práce.

5.2 Sandbox pro útoky DDoS

Útoky Distributed Denial of Service (DDoS) míří na odmítnutí služeb ze strany serveru za vyčerpání výpočetních, paměťových zdrojů, vyčerpání kapacity dané linky nebo mířící na zranitelnost protokolu. Cílem DDoS je vyřazení konkrétního serveru nebo služeb z provozu pro právoplatné uživatele. Skládá se z několika online zařízení připojených do sítě, botnet, které jsou využity pro zahlcení cílového serveru falešným provozem. [94]

Sandbox pro DDoS je určen pro realizaci útoku na webovou stránku. Tím pádem se mohla použít část předešlého sandboxu na jeho vytvoření. Na vytvoření sandboxu byly použity tři virtuální stroje ze stejného vagrant boxu a se stejnou IP adresou a stejnou síťovou topologií jak to bylo u sandboxu pro ssl-strip. *Sandbox.yml* obsahoval také stejné parametry jako u ssl-stripu. Jedinou změnou byla změna názvu webového serveru na **web**, změna názvu VM **victim** na **user** a tím i změna názvu sítí. Po vytvoření Vagrantfilu následovala dodatečná konfigurace pomocí Ansible, kde byly konfigurovány stroje **web** a **attacker**.

Konfigurace VM web

Jako první byl nakonfigurován VM **web**. Konfigurační soubory byly uloženy ve složce *provisioning/roles/apache2/tasks*, kde se vytvořil seznam požadovaných úkolů, které byly rozděleny do souboru dle vykonávané oblasti. Do hlavního playbooku *main.yml* byly vloženy odkazy na tyto soubory: *apache2.yml, set_page.yml, restart_apache2.yml, nload.yml*.

V souboru *apache2.yml* byly uloženy příkazy pro instalaci webového serveru *apache2* a nahrán konfigurační soubor pro webovou stránku.

Soubor *set_page.yml* zajišťuje nahrání souborů pro webovou stránku. Byl vytvořen konfigurační soubor pro virtuálního hosta, který byl následně nahrán na server. Obsahoval naslouchající port pro webovou stránku, jméno serveru, kořenový adresář a ukládání chybových hlášek. Nastavením portu zůstalo ve výchozím stavu, tzn. port 80. Dále byla vytvořena jednoduchá webová stránka obsahující rozsáhlejší obsah. Pro názornost útoku DDoS byl zvolen velký obrázek.

Soubor *nload.yml* obsahuje příkaz na instalaci nástroje *nload* pro monitorování aktuálního příchozího a odchozího provozu byl nainstalován nástroj nload [95].

Zapnutí virtuálního hostu pro webovou stránku, přidání pravidla pro firewall a jeho zapnutí a kontrolu syntaxe pro *apache2*, bylo nastaveno pomocí souboru *restart_apache2.yml*.

Konfigurace VM attacker

Na virtuální stroji **attacker** byly staženy nástroje potřebné pro úspěšnou realizaci útoku. Z již nainstalovaných nástrojů byl vybrán nástroj pro testování sítí a hostitelů *hping3*, který je využitelný pro záplavové útoky [96]. Nástroj *slowLoris* pro logické útoky byl naklonován z githubu a následně nainstalován.

Všeobecná konfigurace

Pro sandbox DDoS bylo taktéž nutné provést dodatečnou úpravu ve Vagrantfilu. Bylo prohozeno pořadí VM na: **br, R, web, attacker** a **user**. Velikost paměti RAM u VM nebyla měněna.

Následně byla vytvořena laboratorní úloha s popisem útoku a pro jeho obranu. Další částí úlohy je vytvořený a odzkoušený scénář pro červený tým a pro modrý tým pro realizaci vybraných útoků, na který byl sandbox vytvořen, viz příloha A.

Závěr

Cílem bakalářské práce bylo navrhnout a implementovat vhodné prostředí pro provedení bezpečnostních cvičení. V teoretické části byly popsány dostupné virtualizační nástroje pro jednotlivé typy virtualizace: virtualizace vázaná na HW, virtualizace na úrovni operačního systému - kontejnerizace, virtualizace využívající více HW zařízení - cloud computing.

Byly charakterizovány nástroje jako jsou VirtualBox, VMware, Docker, Linux containers a OpenStack. V rámci OpenStacku byla popsána architektura a krátce byly popsány možnosti nasazení: all-in-one nebo multinode. Dále byly popsány a srovnány vybrané možnosti instalace, například DevStack, OpenStack-Helm nebo Kolla. V poslední kapitole teoretické části byly popsány vybrané tréningové platformy využívající virtualizační technologie. Na začátku praktické části byla uvedena souvislost mezi vybranou platformou a KYPO projektem. Dále byla popsána a zprovozněna vybraná virtualizační platforma sandbox-creator, která je kompatibilní s KYPO projektem a využívá OpenStack v rámci vytváření souborové struktury pro sandbox. Následně byly charakterizovány konfigurační nástroje využívané v průběhu realizace praktické části práce.

Pro vytvoření prostředí byly pečlivě vybrány témata pro bezpečnostní cvičení. Pak bylo vytvořeno prostředí pro dva bezpečnostní cvičení typu Red/Blue, které byly uvedeny v přílohách A, B. Dále byly vytvořeny laboratorní cvičení s popisem problematiky a vyzkoušeným scénářem pro jednotlivé týmy.

Literatura

[1] *Penetrační test (Penetration test)* [online].[cit. 5. 03. 2021]. Dostupné z URL: <https://managementmania.com/cs/penetracni-test>.

[2] TOVARŇÁK D. *KYPO Cyber Range Design and Use Cases, ICSOFT CONFERENCE 2017* [online].[cit. 10. 02. 2020]. Dostupné z URL: <https://is.muni.cz/publication/1386573/2017-ICSOFT-kypo-cyber-range-design-presentation.pdf>.

[3] AZAM, Noor a Muhammad HARITH bin a BEURAN a RAZVAN. *Usability Evaluation of Open Source and Online Capture the Flag Platforms* [online].[cit. 10. 02. 2020]. Dostupné z URL: <https://www.redhat.com/en/topics/virtualization/what-is-virtualization>.

[4] *King of the Hill: A Novel Cybersecurity Competition for Teaching Penetration Testing* [online].[cit. 10. 02. 2020]. Dostupné z URL: <https://koth.cs.umd.edu/>.

[5] *Network King of the Hill: The fun and lazy hacker CTF* [online].[cit. 10. 02. 2020]. Dostupné z URL: <https://www.itnews.com.au/news/network-king-of-the-hill-the-fun-and-lazy-hacker-ctf-351094>.

[6] *Train your teams against Cyber Attack Threats* [online].[cit. 10. 02. 2020]. Dostupné z URL: <https://cybertestsystems.com/#!/cyber-range>.

[7] VYKOPAL, J. a M. VIZVÁRY a R. OŠLEJŠEK a P. ČELEDA a D. TOVARŇÁK. Lessons learned from complex hands-on defence exercises in a cyber range IN *2017 IEEE Frontiers in Education Conference (FIE)* [online].[cit. 10. 02. 2020]. 1-8 stran. IEEE, 2017. Dostupné z URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8190713>.

[8] *Cybersecurity Red Team Versus Blue Team — Main Differences Explained* [online].[cit. 10. 02. 2020]. Dostupné z URL: <https://securitytrails.com/blog/cybersecurity-red-blue-team>.

[9] *What is virtualization?* [online].[cit. 10. 02. 2020]. Dostupné z URL: <https://www.redhat.com/en/topics/virtualization/what-is-virtualization>.

[10] *Co je virtualizace?* [online].[cit. 10. 02. 2020]. Dostupné z URL: <https://azure.microsoft.com/cs-cz/overview/what-is-virtualization/>.

[11] *Edge Computing Virtualization* [online].[cit. 10. 02. 2020]. Dostupné z URL: <https://www.sdxcentral.com/edge/definitions/mec-virtualization/>.

[12] STODŮLKA, Tomáš. *Platforma pro virtualizaci komunikační infrastruktury 2019* [online].[cit. 10. 02. 2021]. Dostupné z URL: <https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=209359>.

[13] EDER, Michael. *Hypervisor- vs. Container-based Virtualization* [online]. 2016 [cit. 5. 03. 2021]. Dostupné z URL: <https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2016-07-1/NET-2016-07-1_01.pdf>.

[14] *Containerization vs. Virtualization: What's the Difference?* [online].[cit. 5. 03. 2021]. Dostupné z URL: <https://www.burwood.com/blog-archive/containerization-vs-virtualization>.

[15] *Best virtual machine software of 2021: virtualization for different OS* [online].[cit. 28. 02. 2020]. Dostupné z URL: <https://www.techradar.com/best/best-virtual-machine-software>.

[16] *Hardware Virtualization- The New Trend In The Industry of Servers* [online].[cit. 10. 02. 2020]. Dostupné z URL: <https://www.rackbank.com/blog/hardware-virtualization-new-trend-industry-servers/>.

[17] *What Is A Hypervisor? Types Of Hypervisors 1 2* [online].[cit. 3. 03. 2021]. Dostupné z URL: <https://phoenixnap.com/kb/what-is-hypervisor-type-1-2>.

[18] *Xen.org History* [online].[cit. 3. 03. 2021]. Dostupné z URL: <http://www-archive.xenproject.org/community/xenhistory.html>.

[19] *A performance analysis of Xen and KVM hypervisors for hosting the Xen Worlds Project* [online].[cit. 3. 03. 2021]. Dostupné z URL: <https://lib.dr.iastate.edu/cgi/viewcontent.cgi?article=3243&context=etd>.

[20] DESAI, Ankita a OZA Rachana a SHARMA Pratik a PATEL Bhautik. *Hypervisor: A Survey on Concepts and Taxonomy* [online].[cit. 3. 03. 2021]. Dostupné z URL: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.676.380&rep=rep1&type=pdf>.

[21] *THE HYPERVISOR (X86 ARM)* [online].[cit. 3. 03. 2021]. Dostupné z URL: <https://xenproject.org/developers/teams/xen-hypervisor/>.

[22] *Supported Windows guest operating systems for Hyper-V on Windows Server* [online].[cit. 3. 03. 2021]. Dostupné z URL: <https://docs.microsoft.com/en-us/windows-server/virtualization/hyper-v/supported-windows-guest-operating-systems-for-hyper-v-on-windows>.

[23] *What Is Hyper V: The Authoritative Guide* [online].[cit. 3. 03. 2021]. Dostupné z URL: <https://www.acronis.com/en-us/articles/hyper-v-authoritative-guide/>.

[24] *VMware Workstation Player vs VMware Workstation Pro* [online].[cit. 28. 02. 2020]. Dostupné z URL: <https://www.vembu.com/blog/vmware-workstation-player-vs-vmware-workstation-pro/>.

[25] MILUTINOVIĆ, Nikola a Nikola POPOVIĆ. *Posebne pogodnosti u radu sa perifernim uredjajima u VMware okruţenju* [online].[cit. 10. 02. 2020]. Dostupné z URL: <http://2009.telfor.rs/files/radovi/10_43.pdf>.

[26] HASHIMOTO M. *Vagrant: Up and Running: Create and Manage Virtualized Development Environments* [online].[cit. 5. 03. 2021]. Dostupné z URL: <https://books.google.cz/books?hl=cs&lr=&id=7rJqqKCvdagC&oi=fnd&pg=PR2&dq=vagrant+guest&ots=qDwHXwcTD0&sig=A189ajqTTNvuWaSyAVC3nCMVVU8&redir_esc=y#v=onepage&q=vmware&f=false>.

[27] *Oracle VM Virtual Box* [online].[cit. 5. 03. 2021]. Dostupné z URL: <https://viser.edu.rs/uploads/2018/03/7.%20Predavanje%20-%202.pdf>.

[28] *Welcome to VirtualBox.org!* [online].[cit. 10. 02. 2020]. Dostupné z URL: <https://www.virtualbox.org/>.

[29] *Parallels - About* [online].[cit. 2. 03. 2020]. Dostupné z URL: <https://www.crunchbase.com/organization/parallels>.

[30] *Parallels Desktop for Mac* [online].[cit. 2. 03. 2020]. Dostupné z URL: <https://searchvirtualdesktop.techtarget.com/definition/Parallels-Desktop-for-Mac>.

[31] PANDEY, Rachit. *Comparing VMware Fusion, Oracle VirtualBox, Parallels Desktop implemented as Type-2 hypervisors* [online].[cit. 5. 03. 2021]. Dostupné z URL: <https://www.researchgate.net/profile/Rachit-Pandey-3/publication/344046461_Comparing_VMware_Fusion_Oracle_VirtualBox_Parallels_Desktop_implemented_as_Type-2_hypervisors/links/5f4fbf75a6fdcc9879c18621/Comparing-VMware-Fusion-Oracle-VirtualBox-Parallels-Desktop-implemented-as-Type-2-hypervisors.pdf>.

[32] *What are the Virtual Machine hardware limits?* [online].[cit. 2. 03. 2020]. Dostupné z URL: <https://kb.parallels.com/120658>.

[33] BELLARD, Fabrice. *QEMU, a Fast and Portable Dynamic Translator* [online].[cit. 5. 03. 2021]. Dostupné z URL: <https://www.usenix.org/legacy/event/usenix05/tech/freenix/full_papers/bellard/bellard.pdf>.

[34] *What is QEMU?* [online].[cit. 5. 03. 2021]. Dostupné z URL: <https://www.qemu.org/>.

[35] *Main Page* [online].[cit. 5. 03. 2021]. Dostupné z URL: <https://wiki.qemu.org/Main_Page>.

[36] MAGNUS, Jan a Granberg OPSAHL. *Master thesis: Open-source virtualization Functionality and performance of Qemu/KVM, Xen,Libvirt and VirtualBox* [online].[cit. 5. 03. 2021]. Dostupné z URL: <https://www.duo.uio.no/bitstream/handle/10852/37427/Opsahl_Master.pdf?sequence=1&isAllowed=y>.

[37] *Virtualization via Containers* [online].[cit. 10. 02. 2020]. Dostupné z URL: <https://insights.sei.cmu.edu/sei_blog/2017/09/virtualization-via-containers.html>.

[38] *Docker overview* [online].[cit. 10. 02. 2020]. Dostupné z URL: <https://docs.docker.com/engine/docker-overview/>.

[39] Babak Bashari RAD a Harrison John BHATTI a Mohammad AHMADI. *An Introduction to Docker and Analysis of its Performance* [online].[cit. 5. 03. 2021]. Dostupné z URL: <https://d1wqtxts1xzle7.cloudfront.net/52736106/IJCSNS-20170327.pdf?1492765779=&response-content-disposition=inline%3B+filename%3DAn_Introduction_to_Docker_and_Analysis_o.pdf&Expires=1618321349&Signature=Y0Larz6LdYeYdhCgPlRV92YPkRy9vhU~hjDwxwz-uNUtSI6hjAYPA2~GTjszXcva3nS5-c~IiGoE13NEuDVYe8dUmffjyLnDCO2nSwZaH5nEmO-~yCWaz0~NN0Nj6AV-CGcD4kDottSgngDT7tGvR9DuoAtYZiOR1sfU20VKEZRa0bHvN1QAqT0~zQGkLozw07lTbAZ-lX3CCvHmKe-KpeIiO7iyO7A3VEbm3UFCSbZCGVVvkpbrodV0mD3ZwDWHWZg0NUNwp_&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA>.

[40] *Docker(software)* [online].[cit. 10. 02. 2020]. Dostupné z URL: <https://github.com/moloch--/RootTheBox/wiki/Docker-Deployment>.

[41] *Docker* [online].[cit. 10. 02. 2020]. Dostupné z URL: <https://searchitoperations.techtarget.com/definition/Docker>.

[42] JOY, A. M. 2015 International Conference on Advances in Computer Engineering and Applications *Performance comparison between Linux containers and virtual machines* [online]. 2015, poslední aktualizace 23. 7. 2015 [cit. 5. 03. 2021]. Dostupné z URL: <https://ieeexplore.ieee.org/abstract/document/7164727/citations>.

[43] *LXC brief introduction and use* [online].[cit. 10. 02. 2021]. Dostupné z URL: <https://www.programmersought.com/article/9659181034/>.

[44] *What's LXC?* [online].[cit. 10. 02. 2020]. Dostupné z URL: <https://linuxcontainers.org/lxc/introduction/>.

[45] *LXC: features, pros, and cons of Linux Containers* [online].[cit. 10. 02. 2021]. Dostupné z URL: <https://www.ionos.com/digitalguide/server/know-how/what-is-lxc-linux-containers/>.

[46] *What is Kubernetes?* [online].[cit. 10. 02. 2021]. Dostupné z URL: <https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/>.

[47] *Co je cloud computing?* [online].[cit. 10. 02. 2020]. Dostupné z URL: <https://azure.microsoft.com/cs-cz/overview/what-is-cloud-computing/>.

[48] HÖFER, C.N. a KARAGIANNIS G. *Cloud computing services: taxonomy and comparison* [online].[cit. 5. 03. 2021]. Dostupné z URL: <https://link.springer.com/content/pdf/10.1007/s13174-011-0027-x.pdf>.

[49] *What's the difference between cloud and virtualization?* [online].[cit. 10. 02. 2020]. Dostupné z URL: <https://www.redhat.com/en/topics/cloud-computing/cloud-vs-virtualization>.

[50] *Best Free and Open Source IaaS Software* [online].[cit. 3. 03. 2021]. Dostupné z URL: <https://www.linuxlinks.com/iaas/>.

[51] *Cloud Computing Architecture* [online].[cit. 10. 02. 2021]. Dostupné z URL: <https://www.javatpoint.com/cloud-computing-architecture>.

[52] MILOJIČIĆ, D. a I. M. LLORENTE a R. S. MONTERO. IEEE Internet Computing *OpenNebula: A Cloud Management Tool* [online]. 2011, poslední aktualizace 17. 3. 2011 [cit. 5. 03. 2021]. Dostupné z URL: <https://ieeexplore.ieee.org/abstract/document/5731584>.

[53] DONEVSKY, Aleksandar a Sasko RISTOV a Marjan GUSEV. Conference: 48th Int. Scientific Conf. on Information, Communication and Energy Systems and Technologies, ICEST 2013 *Comparison of Open Source Cloud Platforms* [online]. 2013 [cit. 3. 03. 2021]. Dostupné z URL: <https://www.researchgate.net/publication/258673883_Comparison_of_Open_Source_Cloud_Platforms>.

[54] *New Edge Cloud Architecture* [online].[cit. 3. 03. 2021]. Dostupné z URL: <https://opennebula.io/>.

[55] KUMAR, Rakesh a Kanishk JAIN a Hitesh MAHARWAL a Neha JAIN a Anjali DADHICH. *Apache CloudStack: Open Source Infrastructure as a Service Cloud Computing Platform. 111-116* [online]. 2 Červenec 2014 [cit. 3. 03. 2021].

Dostupné z URL:
<https://www.researchgate.net/profile/Rakesh-Kumar-34/publication/264397334_Apache_CloudStack_Open_Source_Infrastructure_as_a_Service_Cloud_Computing_Platform/links/53db6b5d0cf2a19eee8b75b8/Apache-CloudStack-Open-Source-Infrastructure-as-a-Service-Cloud-Computing-Platform.pdf>.

[56] *Concepts and Terminology* [online].[cit. 3. 03. 2021]. Dostupné z URL:
<http://docs.cloudstack.apache.org/projects/archived-cloudstack-getting-started/en/latest/concepts.html#deployment-architecture-overview>.

[57] *About CloudStack* [online].[cit. 3. 03. 2021]. Dostupné z URL:
<https://cloudstack.apache.org/>.

[58] *Welcome to Apache CloudStack's Documentation* [online].[cit. 3. 03. 2021]. Dostupné z URL:
<http://docs.cloudstack.apache.org/en/latest/>.

[59] KUMAR, Rakesh a Neha GUPTA a Shilpi CHARU a Kanishk JAIN a Sunil Kumar JANGIR. International Journal of Computer Science and Mobile Computing *Open Source Solution for Cloud Computing Platform Using OpenStack. IJCSMC, Vol. 3, Issue. 5, May 2014, pg.89 – 98* [online]. 2014 [cit. 10. 02. 2020]. Dostupné z URL:
<https://www.researchgate.net/profile/Rakesh-Kumar-34/publication/263581733_Open_Source_Solution_for_Cloud_Computing_Platform_Using_OpenStack/links/0c96053b4be8a8d6f1000000/Open-Source-Solution-for-Cloud-Computing-Platform-Using-OpenStack.pdf>.

[60] *Guide for Interworking Between HUAWEI CloudFabric Solution and Redhat OpenStack* [online].[cit. 10. 02. 2020]. Dostupné z URL:
<https://support.huawei.com/enterprise/en/doc/EDOC1100072313/c5578416/red-hat-openstack-components>.

[61] *What is orchestration?* [online].[cit. 10. 02. 2020]. Dostupné z URL:
<https://www.redhat.com/en/topics/automation/what-is-orchestration>.

[62] *Lifecycle management* [online].[cit. 3. 03. 2021]. Dostupné z URL:
<https://www.openstack.org/software/project-navigator/deployment-tools>.

[63] *DevStack* [online].[cit. 10. 02. 2020]. Dostupné z URL: <https://docs.openstack.org/devstack/latest/>.

[64] *OpenStack Deployment on Ubuntu 18.04 with DevStack* [online].[cit. 10. 02. 2020]. Dostupné z URL: <https://computingforgeeks.com/openstack-deployment-on-ubuntu-with-devstack/>.

[65] *DevStack* [online].[cit. 3. 03. 2021]. Dostupné z URL: <https://docs.openstack.org/devstack/latest/>.

[66] *What is Kubernetes?* [online].[cit. 3. 03. 2021]. Dostupné z URL: <https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/>.

[67] *An Introduction to Helm, the Package Manager for Kubernetes* [online].[cit. 3. 03. 2021]. Dostupné z URL: <https://www.digitalocean.com/community/tutorials/an-introduction-to-helm-the-package-manager-for-kubernetes>.

[68] *Welcome to OpenStack-Helm's documentation!* [online].[cit. 3. 03. 2021]. Dostupné z URL: <https://docs.openstack.org/openstack-helm/latest/>.

[69] *OpenStack Quick start* [online].[cit. 10. 02. 2020]. Dostupné z URL: <https://docs.openstack.org/kolla-ansible/latest/user/quickstart.html>.

[70] *How to install multi node openstack on virtualbox with packstack on CentOS 7* [online].[cit. 10. 02. 2020]. Dostupné z URL: <https://www.golinuxcloud.com/install-openstack-virtualbox-packstack-centos/#Configure_Network_to_install_multi_node_OpenStack_on_VirtualBox>.

[71] *Packstack: Create a proof of concept cloud* [online].[cit. 5. 03. 2021]. Dostupné z URL: <https://www.rdoproject.org/install/packstack/>.

[72] *OpenStack All-in-one via Packstack, a tutorial.* [online].[cit. 5. 03. 2021]. Dostupné z URL: <https://www.reddit.com/r/homelab/comments/6ox07i/openstack_allinone_via_packstack_a_tutorial/>.

[73] *Welcome to the OpenStack Charm Guide* [online].[cit. 5. 03. 2021]. Dostupné z URL:
<https://docs.openstack.org/charm-guide/latest/?_ga=2.16797119.740851151.1615201801-917751084.1614455134>.

[74] *Packstack* [online].[cit. 10. 02. 2020]. Dostupné z URL:
<https://wiki.openstack.org/wiki/Packstack>.

[75] *OpenStack Releases* [online]. Poslední aktualizace 14. dubna 2021 [cit. 10. 02. 2020]. Dostupné z URL:
<https://releases.openstack.org/>.

[76] *Overview OpenStack* [online].[cit. 10. 02. 2020]. Dostupné z URL:
<https://docs.openstack.org/newton/install-guide-ubuntu/overview.html#figure-hwreqs>.

[77] VYKOPAL, Jan a Radek OŠLEJŠEK a Pavel CELEDA a Martin VIZVÁRY a Daniel TOVARŇÁK. *KYPO Cyber Range: Design and Use Cases. 310-321. 10.5220/0006428203100321.* [online]. 2017 [cit. 10. 02. 2020]. Dostupné z URL:
<https://www.researchgate.net/publication/318870639_KYPO_Cyber_Range_Design_and_Use_Cases>.

[78] *EDURange* [online].[cit. 10. 02. 2020]. Dostupné z URL:
<http://www.edurange.org/docs.html>.

[79] *KYPO PLATFORMA* [online].[cit. 10. 02. 2020]. Dostupné z URL:
<https://www.kypo.cz/>.

[80] *KYPO Cyber Range* [online].[cit. 10. 02. 2020]. Dostupné z URL:
<https://crp.kypo.muni.cz/>.

[81] *Vagrant* [online].[cit. 10. 02. 2020]. Dostupné z URL:
<https://www.vagrantup.com/intro/getting-started/>.

[82] *Vagrant vs Docker* [online].[cit. 10. 02. 2020]. Dostupné z URL:
<https://www.slideshare.net/jchase50/vagrant-vs-docker>.

[83] *Boxes* [online].[cit. 20. 10. 2020]. Dostupné z URL:
<https://www.vagrantup.com/docs/boxes>.

[84] *Boxes* [online].[cit. 20. 10. 2020]. Dostupné z URL:
<https://www.zdrojak.cz/clanky/uvod-do-vagrantu/>.

[85] *OVERVIEW HOW ANSIBLE WORKS* [online].[cit. 10. 02. 2020]. Dostupné z URL:
<https://www.ansible.com/overview/how-ansible-works>.

[86] *ANSIBLE IN DEPTH* [online].[cit. 5. 03. 2021]. Dostupné z URL:
<https://cdn2.hubspot.net/hub/330046/file-480366556-pdf/pdf_content/Ansible_in_Depth.pdf?t=1390852822000>.

[87] HALL, Daniel. *Ansible Configuration Management* [online].[cit. 5. 03. 2021]. Dostupné z URL:
<https://books.google.cz/books?hl=cs&lr=&id=ETQmAgAAQBAJ&oi=fnd&pg=PT5&dq=ansible&ots=CKrYwf1WoD&sig=Oama3ZYPqvnMius5a4WZ43dh90Y&redir_esc=y#v=onepage&q&f=false>.

[88] *Getting Started with CTFd* [online].[cit. 10. 02. 2020]. Dostupné z URL:
<https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html>.

[89] *DevOps101-Improve Your Workflow! First Steps on Vagrant* [online].Poslední aktualizace 26. září 2018 [cit. 5. 03. 2021]. Dostupné z URL:
<https://hackernoon.com/devops101-vagrant-6737c8c29904>.

[90] *Download VirtualBox for Linux Hosts* [online].[cit. 10. 02. 2020]. Dostupné z URL:
<https://www.virtualbox.org/wiki/Linux_Downloads>.

[91] *Installing VirtualBox 6.0 on Ubuntu 18.04 LTS /18.10* [online].[cit. 09. 02. 2021]. Dostupné z URL:
<https://blog.exxactcorp.com/installing-virtualbox-6-0-on-ubuntu18-04-lts-18-10/>.

[92] *How to Install Vagrant on Ubuntu 18.04* [online].[cit. 10. 02. 2020]. Dostupné z URL:
<https://linuxize.com/post/how-to-install-vagrant-on-ubuntu-18-04/>.

[93] *SSL/TLS Strong Encryption: How-To* [online].[cit. 29. 11. 2020]. Dostupné z URL:
<https://httpd.apache.org/docs/trunk/ssl/ssl_howto.html>.

[94] *Distributed Denial of Service (DDoS)* [online].[cit. 30. 11. 2020]. Dostupné z URL:
<https://www.imperva.com/learn/ddos/denial-of-service/>.

[95] *nload – Monitor Linux Network Bandwidth Usage in Real Time* [online].[cit. 30. 11. 2020]. Dostupné z URL: <https://www.tecmint.com/nload-monitor-linux-network-traffic-bandwidth-usage/>.

[96] *hping3 Package Description* [online].[cit. 30. 11. 2020]. Dostupné z URL: <https://tools.kali.org/information-gathering/hping3>.

[97] RAJJ, Jakub. *FILTRACE DISTRIBUOVANÝCH ÚTOKŮ NA ODEPŘENÍ SLUŽEB POMOCÍ SÍŤOVÝCH PRVKŮ MIKROTIK 2019* [online].[cit. 3. 5. 2021]. Dostupné z URL: <https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=192593>.

[98] *DDoS attacks and defense mechanisms: classification and state-of-the-art* [online]. 5. duben 2004 [cit. 1. 12. 2020]. Dostupné z URL: <https://www.sciencedirect.com/science/article/abs/pii/S1389128603004250>.

[99] Wong Onn Chee a BRENNAN Tom. *H.....t.....t....p....p....o....s....t* [online]. 11. listopad 2010 [cit. 1. 12. 2020]. Dostupné z URL: <https://owasp.org/www-pdf-archive/Layer_7_DDOS.pdf>.

[100] SCOUT, Net. *What is an ICMP Flood Attack? 2017-03-12* [online].[cit. 2. 12. 2020]. Dostupné z URL: <https://www.netscout.com/what-is-ddos/icmp-flood>.

[101] BOGDANOSKI, M. a SHUMINOSKI T. a RISTESKI A. *Analysis of the SYN Flood DoS Attack* [online]. Červen 2013 [cit. 2. 12. 2020]. Dostupné z URL: <http://eprints.ugd.edu.mk/6729/1/IJCNIS-V5-N8-1.pdf>.

[102] *hping3 Package Description* [online].[cit. 12. 12. 2020]. Dostupné z URL: <https://tools.kali.org/information-gathering/hping3>.

[103] *hping3 flood ddos* [online].[cit. 12. 12. 2020]. Dostupné z URL: <https://linuxhint.com/hping3/>.

[104] YALTIRAKLI, Gokberk. *Slowloris 2015* [online].[cit. 12. 12. 2020]. Dostupné z URL: <https://github.com/gkbrk/slowloris>.

[105] *ICMP (Internet Control Message Protocol)* [online].[cit. 5. 3. 2021]. Dostupné z URL: <http://www.rhyshaden.com/icmp.htm>.

[106] *Apache Module mod_reqtimeout* [online].[cit. 16. 12. 2020]. Dostupné z URL: <https://httpd.apache.org/docs/trunk/mod/mod_reqtimeout.html>.

[107] HODGES, J. a JACKSON C. a BARTH A. *HTTP Strict Transport Security (HSTS)* [online]. 2012 [cit. 5. 3. 2021]. Dostupné z URL: <https://www.hjp.at/doc/rfc/rfc6797.html>.

[108] FIELDING, R. a IRVINE UC a GETTYS J. a MOGUL J. a FRYSTYK H. a MASINTER L. a LEACH P. a BERNERS-LEE T. *Hypertext Transfer Protocol – HTTP/1.1* [online]. 1999 [cit. 5. 3. 2021]. Dostupné z URL: <https://www.rfc-editor.org/rfc/pdfrfc/rfc2616.txt.pdf>.

[109] *Evolution of HTTP* [online].[cit. 3. 5. 2021]. Dostupné z URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP>.

[110] FREIER, A. a KARLTON P. a KOCHER P. *The Secure Sockets Layer (SSL) Protocol Version 3.0* [online]. Srpen 2011 [cit. 5. 3. 2021]. Dostupné z URL: <https://www.hjp.at/doc/rfc/rfc6101.html>.

[111] *The Transport Layer Security (TLS) Protocol Version 1.2* [online].[cit. 5. 3. 2021]. Dostupné z URL: <https://tools.ietf.org/html/rfc5246#page-4>.

[112] *What Are SSL Stripping Attacks?* [online].[cit. 2. 12. 2020]. Dostupné z URL: <https://www.venafi.com/blog/what-are-ssl-stripping-attacks>.

[113] *bettercap 2.x: how to install and use in Kali Linux* [online].[cit. 04. 02. 2021]. Dostupné z URL: <https://miloserdov.org/?p=1112>.

[114] *Bettercap Package Description* [online].[cit. 04. 02. 2021]. Dostupné z URL: <https://tools.kali.org/sniffingspoofing/bettercap>.

[115] *Documentation* [online].[cit. 04. 02. 2021]. Dostupné z URL: <https://www.bettercap.org/legacy/>.

[116] *Firefox: Installing Self-Signed certificate on Ubuntu* [online].[cit. 04. 02. 2021]. Dostupné z URL: <https://dev.to/lmillucci/firefox-installing-self-signed-certificate-on-ubuntu-4f11>.

[117] FICCO, Massimo a Francesco PALMIERI. *Leaf: an open-source cybersecurity training platform for realistic edge-IoT scenarios 2019* [online].[cit. 11. 02. 2020]. <https://www.sciencedirect.com/science/article/pii/S1383762118304442>.

Seznam symbolů, veličin a zkratek

ACK Acknowledgment flag
AMI Amazon Machine Image
API Application programming interface
ARP Address Resolution Protocol
AWS Amazon Web Services
CSR Certificate Signing request
CTF Capture the flag
DDoS Distributed Denial of Service
DNS Domain Name System
HSTS HTTP Strict Transport Security
HTTP Hypertext Transfer Protocol
HTTPS Hypertext Transfer Protocol Secure
HW Hardware
ICMP Internet Control Message Protocol
IP Internet Protocol
KVM Kernel-based Virtual Machine
MAAS Metal As A Service
MITM Man In the Middle
NASA National Aeronautics and Space Administration
OCA OpenNebula Cloud API
OS Operační systém
PHP Hypertext Preprocessor
QEMU Quick emulator
REST Representational State Transfer
RHEL Red Hat Enterprise Linux
RST Reset flag v protokolu TCP
SSH Secure Shell
SSL Secure Sockets Layer
SYN Synchronisation flag
SW Software
TCP Transmission Control Protocol
TLS Protokol Transport Layer Security
UDP User Datagram Protocol
VM Virtual Machine
XML Extensible Markup Language
YAML YAML Ain't Markup Language

A Laboratorní úloha DDoS útoky

A.1 Cíl

Cílem laboratorní práce je vyzkoušet útoky mířící na odmítnutí služeb ze strany serveru, DDoS. Budou testovány: záplavové útoky (mířící na vyčerpání kapacity linky) - ICMP Flood a SYN Flood; logický útok (mířící na chybu v protokolu) - SlowLoris. Na druhou stranu bude vyzkoušeno znemožnění nebo zmírnění těchto útoků. Hlavním cílem bude odepření služeb ze strany serveru, tj. nenačtení obsahu webové stránky u uživatele, a získání informací ohledně úspěšnosti útoku nebo použitých protiopatření na straně serveru z odchycených zpráv. Je předpokládáno, že realizace jednotlivých úkolů u Read teamu budou vykryty příslušným úkolem ze strany Blue teamu. To znamená: Read team bude realizovat ICMP flood, Blue team na VM user bude mít spuštěný ping na adresu serveru a na serveru bude reagovat na ICMP flood.

A.2 Teoretický popis

DDoS
Úkolem DDoS je znepřístupnění služeb ze strany serveru pro jiné uživatele. Ve většině případů se posílají zprávy na adresu vybraného cíle, kde způsobí vyčerpání kapacity linky, vyčerpání výpočetních a paměťových zdrojů nebo míří na chybu v protokolu. Dle cíle útoku je můžeme rozdělit na: záplavové útoky, logické útoky a vyčerpání zdrojů serveru [98, 99, 97].

ICMP Flood
Protokol ICMP poskytuje službu pro zjištění dostupnosti daného stroje v síti, ping. Odesílatel posílá dotaz *echo_request* a příjemce odpovídá *echo_reply*. Záplavový útok ICMP flood využívá právě této služby pro vyřazení cíle z provozu. Na IP adresu cíle je zasíláno velké množství *echo_request*, na které stroj odpoví stejným množstvím. Generováním *echo_requestu* je cílový stroj zaneprázdněn a tím není schopen reagovat na další provoz. ICMP flood může být zaměřen na konkrétní počítač nebo server v síti, na router za účelem přerušení komunikaci mezí sítěmi. Hlavní podmínkou realizace útoku je ve znalosti IP adresy cílového stroje.[100]

SYN Flood
Příznak SYN je využit u protokolu transportní vrstvy TCP. Protokol zajišťuje spolehlivý přenos, tzn. navazuje spojení - three-way handshake. Spojení je iniciováno ze strany klienta. Zašle se paket s nastaveným příznakem SYN. Server potvrdí SYN-ACK a klient také potvrdí příznakem ACK. Celkový SYN flood se realizuje zasláním

prvního paketu s příznakem SYN, ve kterém útočník žádá o připojení k serveru. Server v domnění, že se jedná o běžného uživatele, potvrdí připojení a pošle paket se SYN-ACK. V tomto případě se už odpovědi nedočká. Tímto způsobem jsou vytvářena polootevřená spojení až do vyčerpání možnosti vytvořit další. Server čeká na potvrzení připojení ze strany klienta (útočníka) a není schopen obsloužit legitimní uživatele.[101]

SlowLoris

Mezi logické útoky patří Slowloris. K odepření dostupnosti serveru nevyužívá velké množství najednou vygenerovaných paketů. Jeho průběh je pomalejší a tím se jeví jako zcela legitimní provoz. Slowloris zasílá **částečný** HTTP požadavek na server, kde jsou postupně otevírána další a další spojení čekající na dokončení požadavku. Tím se vyčerpá možný počet otevřených spojení a další uživatelé už nejsou obslouženi.

A.3 Praktická realizace

Praktická část je rozdělena na dvě části. První popisuje úkoly pro realizaci hrozby, tzn. pro Red team. Naopak druhá část popisuje úkoly pro zmírnění útoku, pro Blue team. Úloha využívá tři virtuální stroje, které jsou detailněji popsané v tabulce níže, tabulka Tab. A.1.

Poznámka: Po spuštění virtuálních strojů je nutné ověření dostupnosti komunikace mezi nimi pomocí příkazů: *ip route* pro načtení směrovacích cest a provést *ping <IP stanice>* na jednotlivé stroje pro ověření komunikace.

Virtuální stroj-OS	IP adresa	uživ.jméno-heslo
web-ubuntu/xenial64	209.165.200.11	vagrant-vagrant
user-ubuntu16.04	192.168.3.3	ubuntu-vagrant
attacker-kaliLinux	192.168.3.5	vagrant-vagrant

Tab. A.1: Parametry pro používané VM

A.3.1 Red team

Prvním úkolem Red teamu bude vyčerpání kapacity linky pomocí nadměrného počtu *echo_request*, ověření správnosti realizovaného útoku pomocí odchycení komunikace pomocí programu Wireshark a zjištění rozdílu obsahu legitimních a nelegitimních zpráv. Výsledkem ICMP floodu bude delší nebo žádná odezva ze strany serveru pro uživatele. Druhým úkolem Red teamu je ztížit nebo znemožnit načtení webové stránky ze serveru pro uživatele, ověřit útok a zjistit rozdíl odezvy serveru před a po

nastavení omezení pro SYNflood pomocí programu Wireshark. K tomuto účelu bude použit SYN flood.

Třetím úkolem bude vyzkoušení logického útoku slowloris pomocí stejnojmenného nástroje. Nástroj *slowloris* je napsaný v jazyce Python. Po spuštění je vytvořeno velké množství neúplných HTTP požadavků, které jsou následně v 15 sekundovém intervalu zasílány na konkrétní server. Spojení nejsou nikdy uzavřeny a tím server nedokáže odpovídat na jiné dotazy. [104] Pro ověření útoku bude také použit Wireshark, kde bude cílem zjistit, jestli je na serveru nastaveno omezení a pokud ano, tak jakým způsobem je komunikace omezena (například konkrétní typ HTTP zprávy ukončující spojení).

K prvním dvěma záplavovým útokům bude použit nástroj *hping3*. Nástroj byl v minulosti používán jako nástroj pro testování bezpečnosti. Například testování firewallu, sítě, síťových prvků nebo skenování portů. Je orientován na model TCP/IP, kde podporuje protokoly využívané tímto modelem, například UDP, TCP a ICMP. [102] V rámci testování sítě je možné vytvářet pakety s různými parametry. Například vytvoření paketu s danou velikostí, určení množství vytvořených paketů pro vyčerpání zdrojů sítě nebo síťového prvku. [103]

Red team má přístup pouze k VM útočníka, odkud budou realizovány úkoly. Práce bude probíhat pomocí terminálu a ve Wiresharku.

Záplavové útoky

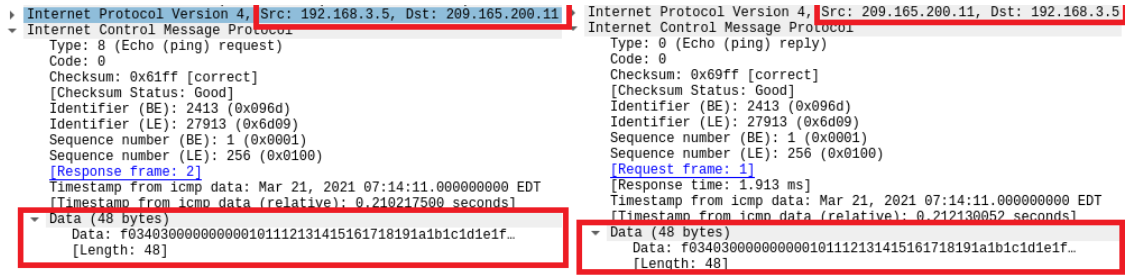
Prvním úkolem je realizace útoku ICMP flood. K útoku bude použit výše popsaný nástroj *hping3* a k jeho ověření program Wireshark. Otevřeme si terminál a přepneme se do role roota pomocí příkazu: *sudo -i*. Dále spustíme program Wireshark a zapneme zachytávání paketů na síťovém rozhraní *eth1*.

V terminálu spustíme ping na webový server příkazem: *ping 209.165.200.11*, který po několika sekundách vypneme. Ve Wiresharku analyzujeme zachycené zprávy, jak *echo_request*, tak *echo_reply*. Po otevření zprávy pro žádost a odpověď v části pro ICMP protokol v hlavičce vidíme typ zprávy, sekvenční číslo nebo rychlost odezvy. Datová část obsahuje identická data v obou zprávách, viz Obr. A.1. Tato část je nepovinná, ale v případě programu **ping** je vyplněna a odesílatel musí vložit identický řetězec do tohoto pole [105].

Ve Wiresharku nastavíme nové zachytávání. Pak se přepneme do terminálu a pomocí nástroje *hping3* spustíme generování ICMP dotazů na IP adresu serveru. Server bude vytížen a uživateli nebude zpracován ICMP echo_request nebo bude značně zpomalen.

```
root@attacker:~# hping3 --icmp --flood 209.165.200.11
```

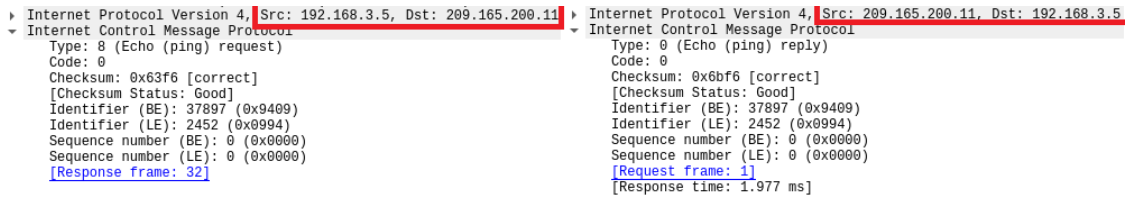
Výpis A.1: Realizace IMCP flood



Obr. A.1: Attacker-zachycené legitimní ICMP pakety

Kde *–icmp* znamená zvolený protokol. *–flood* určuje, že pakety budou zasílány co nejrychleji a nebudou vypsány žádné odpovědi na ně. Následně bude vypsána hláška o generovaných hlavičkách paketů bez datové části. Běh nástroje se ukončuje pomocí Ctrl+C, kde se ukáže statistika o odeslaných a přijatých paketech.

Průběh útoku ověříme ve Wiresharku, kde zprávy zachytíme a zobrazíme. ICMP request od attackeru neobsahuje datovou část a tím ani příslušná odpověď od serveru neobsahuje datovou část, viz Obr. A.3.



Obr. A.2: Attacker-zachycené nelegitimní ICMP pakety

Druhý útok mířený na vyčerpání služeb je SYN flood. V našem případě budeme cílit na konkrétní port pro HTTP protokol, 80. Před spuštěním útoku spustíme zachytávání komunikace ve Wiresharku. Útok spustíme pomocí *hping3*:

```
root@attacker:~# hping3 -S --flood -V -p 80 209.165.200.11
```

Výpis A.2: Realizace SYN flood na port 80

Parametr *-S* určuje, že se budou posílat pakety s příznakem SYN. Parametr *-p 80* určuje vybraný port dané služby, konkrétně služba přenosu dat internetem s portem 80. Pak následuje IP adresa cíle. Útok znova ověříme ve Wiresharku, kde na začátku jsou odesílány požadavky ze strany útočníka, dále jsou ze strany serveru tyto požadavky potvrzeny s příznakem SYN-ACK. Po krátké době jsou tato spojení resetovány ze strany útočníka s příznakem RST. V případě již nastaveného omezení na straně serveru, je množství odpovědí menší než odeslaných požadavků ze stroje útočníka.

Logické útoky

Nyní bude otestován záplavový útok slowloris, který byl popsán výše. V programu

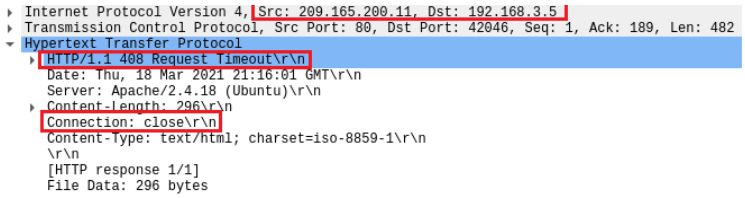
Wireshark spustíme zachytávání provozu. V terminálu se přepneme do složky obsahující nástroj slowloris *cd slowloris* a spustíme:

```
root@attacker:~# python3 slowloris.py -v -s 10000 209.165.200.11
```

Výpis A.3: Spuštění útoku slowloris

Parametr *-v* nastavuje výpis na terminálu, *-s* stanovuje počet zaslaných soketů. Na terminálu se postupně začne vypisovat běh. Prvně jsou vytvořeny pakety a pak jsou tyto pakety zaslány na cíl. Následuje přestávka na 15 sekund a pak se pošle znova další paket.

Odezvu serveru ověříme ve Wiresharku, kde jsou zasílány odpovědi z webového serveru. V případě již nastaveného omezení lze mezi odpovědi najít pakety s typem zprávy *408 Request timeout* a obsahem, který sděluje, že spojení je uzavřené.



Obr. A.3: Attacker-zachycený HTTP paket 408 Request timeout

A.3.2 Blue team

Úkolem Blue týmu bude zmírnit DDoS útoky mířené na webový server a udržení komunikace s uživatelem. Blue team má přístup k virtuálním strojům webového serveru a uživatele. Na VM web budou úkoly prováděny v terminálu a na VM user v terminálu a prohlížeči firefox. K sledování vytížení na serveru bude použit nástroj *nload*, který slouží k monitorování provozu a šířky pásma na dané lince [95]. Nástroj používá příkazový řádek a monitoruje příchozí a odchozí provoz na dané lince, které vykresluje pomocí grafu a přidává také číselné hodnoty.

Monitorování a zmírnění ICMP flood

Na virtuálním stroji web si otevřeme terminál přepneme se do role roota pomocí *sudo -i* a spustíme nástroj *nload*.

```
root@web:~# nload --interface enp0s8 -i 100 -o 100 -u m
```

Výpis A.4: Spuštění nload s danými parametry

Parametr *–interface* určuje vybrané rozhraní, v našem případě enp0s8. *-i* nastavuje kapacitu linky pro příchozí provoz v kbit/s, *-o* nastavuje kapacitu odchozí linky, také v kbit/s. Kapacita linky je záměrně snížena kvůli lepší názornosti laboratorní úlohy.

Parametr *-u m* nastavuje zobrazení jednotek v grafu, konkrétně Mbit/s. Po spuštění se ukáže prázdný graf zvlášť pro příchozí (*Incoming*) a odchozí (*Outgoing*) přenos i s číselnými hodnotami na pravé straně. Po spuštění útoku u Red teamu nástroj *nload* zachytí provoz a zobrazí na grafu. Také hodnota *Curr:* se bude měnit na základě hustoty provozu. V případě hustého provozu by se tato hodnota měla přibližovat nebo téměř rovnat k maximální kapacitě linky.
Přítomnost ICMP flood si můžeme ověřit i jiným způsobem. Ukončíme *nload* a spustíme tcpdump pro sledování námi vybraného rozhraní: *tcpdump –interface enp0s8*. Ve výpisu by se měli objevit po sobě následující echo_requesty ze strany útočníka, Obr. A.4.

```
16:53:17.393651 IP attacker > web: ICMP echo request, id 13073, seq 54959, lengt
h 8
16:53:17.393652 IP attacker > web: ICMP echo request, id 13073, seq 55215, lengt
h 8
16:53:17.393653 IP attacker > web: ICMP echo request, id 13073, seq 55471, lengt
h 8
16:53:17.393654 IP attacker > web: ICMP echo request, id 13073, seq 55727, lengt
h 8
16:53:17.393803 IP attacker > web: ICMP echo request, id 13073, seq 55983, lengt
h 8
16:53:17.393820 IP attacker > web: ICMP echo request, id 13073, seq 56239, lengt
h 8
16:53:17.393821 IP attacker > web: ICMP echo request, id 13073, seq 56495, lengt
h 8
```

Obr. A.4: Běh tcpdump na serveru - ICMP flood

Na VM user v terminálu spustíme ping na adresu webového serveru (209.165.200.11). Bez velkého zatížení linky by server měl odpovídat v řádech desítek milisekund, Obr. A.5. V případě, že bude linka zatížená, server by měl odpovídat ve větších prodlevách, v řádu desítek nebo stovek mili sekund, viz obrázek Obr. A.6.

```
64 bytes from 209.165.200.11: icmp_seq=55 ttl=63 time=1.87 ms
64 bytes from 209.165.200.11: icmp_seq=56 ttl=63 time=1.91 ms
64 bytes from 209.165.200.11: icmp_seq=57 ttl=63 time=2.09 ms
64 bytes from 209.165.200.11: icmp_seq=58 ttl=63 time=0.865 m
```

Obr. A.5: Odezva serveru na straně uživatele

```
64 bytes from 209.165.200.11: icmp_seq=134 ttl=63 time=54.8 ms
64 bytes from 209.165.200.11: icmp_seq=135 ttl=63 time=146 ms
64 bytes from 209.165.200.11: icmp_seq=136 ttl=63 time=238 ms
```

Obr. A.6: Odezva serveru u uživatele - ICMP flood

Zmírnění ICMP flood bude proveden jednoduchým omezením, pomocí *iptables* na straně serveru.

```
root@web:~# iptables -A INPUT -p icmp --icmp-type echo-request
          -m limit --limit 60/minute --limit-burst 120 -j ACCEPT
```

Výpis A.5: Sledování provozu přes tcpdump

Do iptables jsme přidali pravidla: povolení ICMP echo_requestu se shodováním se 120 paketů za jednu minutu. Tím je dosaženo snížení vytížení linky na serveru.

Monitorování a zmírnění SYN flood

Na virtuálním stroji web si zobrazíme pravidla *iptables* příkazem *sudo iptables -L –line-numbers* a vymažeme pravidlo s příslušným číslem pomocí příkazu *sudo iptables -D INPUT <číslo_pravidla>*. Nload znova spustíme. Na VM user spustíme prohlížeč firefox a načteme si stránku: http://209.165.200.11. Při zaneprázdnění serveru by se stránka měla načítat delší dobu nebo by se neměla načíst vůbec. Na VM web zatím pozorujeme graf. V případě spuštění SYN flood by příchozí i odchozí provoz měl být na maximu. Server dostává požadavek na spojení a odpovídá SYN, ACK. Dotazy s příznakem si ještě můžeme zobrazit pomocí *tcpdump –interface enp0s8*. Na straně uživatele pořád zkoušíme načítání webové stránky. SYN flood bude taktéž zmírněn pomocí pravidel v *iptables* na straně serveru.

```
root@web:~# iptables -A INPUT -p tcp -m state --state NEW
          -m recent --update --seconds 30 --hitcount 2 -j DROP
root@web:~# iptables -A INPUT -p tcp -m state --state NEW
          -m recent --set -j ACCEPT
```

Výpis A.6: Přidání pravidel do iptables - SYN flood

První pravidlo zahazuje každý druhý dotaz za 30 sekund. Druhé pravidlo povoluje nové neukončené připojení (hodnota čísel je uvedena pouze pro názornost). V případě, že víme, že útok přichází z jedné IP adresy, můžeme použit omezení počtu připojení na konkrétní IP adresu:

```
root@web:~# iptables -A INPUT -p tcp --syn --dport 80
          -m connlimit --connlimit-above 15 --connlimit-mask 32
          -j REJECT --reject-with tcp-reset
```

Výpis A.7: Omezení připojení na jednu IP adresu

Pravidlo omezuje připojení na 15 z jedené IP adresy.

Monitorování a zmírnění logického útoku Slowloris

Na virtuálním stroji web vymažeme předešle nastavené pravidlo v *iptables* pomocí příkazu *iptables -F*. Prohlížeč firefox s webovou stránkou necháme otevřený. Pro názornost korektního průběhu načtení celé webové stránky v terminálu spustíme **tcpdump –interface enp0s8** a načteme webovou stránku u uživatele. *Tcpdump* zachytil kompletní přenos načtení stránky, pomocí Ctrl+C ukončíme. Nload znovu

spustíme. Na VM web zatím pozorujeme graf. V případě spuštění slowloris ze strany útočníka by odchozí a příchozí provoz měl narůst v určitých časových intervalech. To jsou u útočníka generovány požadavky a zasílány na server v několika sekundových intervalech. Zasílané požadavky od útočníka a odpovědi serveru si také můžeme ověřit pomocí **tcpdump –interface enp0s8**. Na straně uživatele pořád zkoušíme načtení webové stránky. Stránka by se měla načítat dlouhou dobu. Také si ve výpisu tcpdump můžeme ověřit, že požadavky jsou zasílány vždy na jiný port serveru.
Tyto nežádoucí polootevřená připojení můžeme omezit v nastavení webového serveru *apache2*. Kde zapneme modul *reqtimeout*, který nastavuje časový limit a minimální rychlost pro příchozí spojení [106].

```
root@web:~# a2enmod reqtimeout
```

Výpis A.8: Zapnutí modulu reqtimeout

Po zapnutí si otevřeme konfigurační soubor pro tento modul, kde nastavíme časový limit pro příchozí dotazy. (Výchozí hodnotou je čekání na hlavičku od 20 do 40 sekund.) Bude nastaven maximální čas pro přijetí hlavičky zprávy, od 10 do 20 sekund. Dále bude nastaven maximální limit pro přijetí těla zprávy, také od 10 do 20 sekund. Minimální velikost přenesených částí zprávy bude ponechána na 500B. Toto nastavení částečně eliminuje útok slowloris a tím se dokáže připojit k serveru i legitimní uživatel.

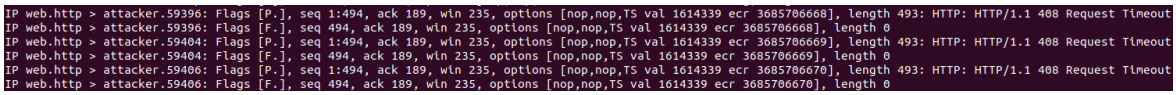
Poznámka:

Naše webová stránka nevyužívá žádné přídavné protokoly pro zabezpečení, jako TLS nebo SSL. To znamená, že není potřeba ověřování certifikátu během připojování a to vyžaduje více času než budeme nastavovat. Proto pro stránky používající zabezpečení se nedoporučuje tyto hodnoty měnit. Mohlo by to způsobit nemožnost připojení u legitimního uživatele.

```
root@web:~# cd /etc/apache2/mods-enabled/
root@web:~# nano reqtimeout.conf
#odkomentujeme řádek a změníme hodnoty
RequestReadTimeout header=10-20,MinRate=500
body=10-20,minrate=500
root@web:~# service apache2 restart
```

Výpis A.9: Nastavení parametrů modulu reqtimeout

Po nastavení limitu by mělo dojít k omezení útoku. Některé pakety nebudou stíhat nastavené intervaly a tím se neotevřou nová polootevřená spojení. V tomto případě server bude zasílat chybovou hlášku *408 REQUEST TIMEOUT* [106]. Po zastavení zachytávání(CTRL+C) v *tcpdump* bychom měli najít několik odeslaných chybových hlášek na stroj útočníka, viz obrázek Obr. A.7.



Obr. A.7: Tcpdump-odeslaná chybová hláška ze serveru

A.4 Závěr

V laboratorní úloze byly na straně červeného týmu odzkoušeny záplavové útoky (ICMP flood a SYN flood) a ověřeny pomocí programu Wireshark. V rámci útoku ICMPflood byly srovnány rozdíly mezi legitimními a nelegitimními zprávami. U útoku SYNflood bylo srovnáno množství zasílaných odpovědí ze serveru před a po nastavení omezení. Také byl odzkoušen logický útok SlowLoris a zjištěno nastavené omezení na serveru pro zmírnění útoku, které bylo zachyceno a zobrazeno programem Wireshark. Na straně blue týmu byly nastaveny protiopatření na výše zmíněné útoky. Protiopatření pro ICMPflood a SYNflood byly nastaveny pomocí nastavených pravidel v *iptables*. Útok SlowLoris byl zmírněn nastavením časových limitů v modulu *reqtimeout* u serveru *apache2*.

B Laboratorní úloha ssl-strip

B.1 Cíl

Cílem laboratorní úlohy je vyzkoušet degradaci zabezpečeného přenosu HTTPS na HTTP, pomocí útoku zaměřeného na SSL protokol, a to ssl-strip. Na druhou stranu si také vyzkoušet znemožnění a současně odvrácení útoku pomocí nastavení přídavného zabezpečení na straně serveru. Hlavním cílem bude získání přihlašovacích údajů z konkrétní webové stránky.

B.2 Teoretický popis

Tato část obsahuje popis dané problematiky. Bude popsán nezabezpečený přenos a jeho nevýhody. K tomu bude popsán zabezpečený přenos. A jako poslední bude popsáno jak obejít zabezpečení a získat tím citlivé informace.

Nezabezpečený přenos - HTTP

Pro přenos webových služeb a aplikací ve formátu HTML, XML nebo PHP se používá Hypertext Transfer Protocol, HTTP. Pro přenos se používá spolehlivý protokol TCP s portem 80, který ale pro přenos souborů nepoužívá šifrování ani nezaručuje neporušení přenášených dat jiným účastníkem během přenosu. [108, 107]

Zabezpečený přenos - HTTPS

Pro eliminaci nevýhod protokolu HTTP se používá protokol HTTP Secure (HTTPS), který využívá protokoly SSL nebo TLS pro zabezpečení. Tím je odstraněna nevýhoda u HTTP a zajištěna autentizace, důvěrnost a integrita přenášených dat. [107]

Protokoly zajišťující bezpečnost přenosu

Protokoly pro zajištění bezpečnosti webového či jiných datových přenosů jsou: starší Secure Sockets Layer(SSL) nebo novější Transport Layer Security(TLS). Tyto protokoly jsou implementovány mezi aplikační a transportní vrstvou modelu TCP/IP. Umožňují důvěrnost a integritu přenášených dat mezi dvěma komunikujícími stranami za použití šifrování. K šifrování je použita symetrická nebo asymetrická kryptografie. V případě webových je použita asymetrická kryptografie v rámci certifikátů. [110, 111]

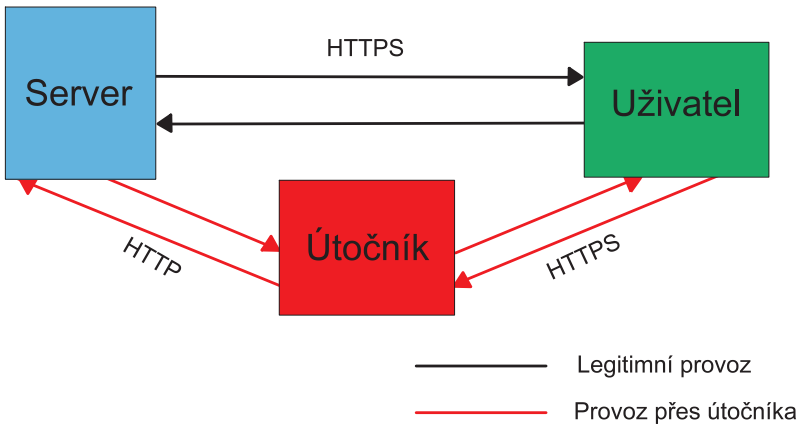
HTTP Strict Transport Security - HSTS

Bezpečnostní mechanismus HSTS sloužící k ochraně komunikace mezi webovým prohlížečem a serverem před degradačními útoky, jako je například SSL-strip [112]. HSTS zajišťuje, aby webový server vynutil u prohlížeče přenos pouze přes zabezpečené připojení, pomocí HTTPS. Mechanismus HSTS je aplikován ze strany serveru, který zasílá informační zprávu klientovi pomocí HTTPS. Zpráva obsahuje oznámení,

že daný web bude používat pouze zabezpečený přenos. Pak na straně klienta proběhne přenastavení všech odkazů pro danou webovou stránku na HTTPS. [107]

SSL-Strip

Poprvé SSL-strip byl představen v roce 2009 americkým počítačovým výzkumníkem Moxiem Marlinspikem na schůzi Black Hatu. Jedná se o nebezpečnou zranitelnost s možností krádeže přihlašovacích údajů uživatelů. Funguje na principu přesměrování HTTPS provozu přes vytvořený kanál útočníka, vznik formace ManInTheMiddle. Následně je zabezpečený provoz mezi útočníkem a uživatelem degradován na nezabezpečený přenos dat bez šifrování, tím, že útočník nepřepošle uživateli redirect na HTTPS. Na straně mezi serverem a útočníkem přenos zůstává ve stejném formátu, zabezpečený. Naopak na straně mezi útočníkem a uživatelem přenos bude nezabezpečený. Tím, že uživatel používá nezabezpečený protokol HTTP útočník je schopen odchytit a využít jeho požadavek na server, viz Obr. B.1. [112]



Obr. B.1: Schéma realizace sslstripu

B.3 Praktická realizace

Praktická část je rozdělena na dvě části. První popisuje úkoly pro realizaci hrozby, tzn. pro Red team. Naopak druhá část popisuje úkoly pro znemožnění a odvrácení možného, dalšího útoku, pro Blue team. Úloha využívá tři virtuální stroje, které jsou detailněji popsané v tabulce níže.

Poznámka: Po spuštění virtuálních strojů je nutné ověření dostupnosti komunikace mezi nimi pomocí příkazů: *ip route* pro načtení směrovacích cest a provést *ping <IP stanice>* mezi jednotlivými stroji pro ověření komunikace.

Virtuální stroj-OS	IP adresa	uživ. jméno-heslo
apache2-ubuntu/xenial64	209.165.200.11	vagrant-vagrant
victim-ubuntu16.04	192.168.3.3	ubuntu-vagrant
attacker-kaliLinux	192.168.3.5	vagrant-vagrant

Tab. B.1: Popis parametrů pro virtuální stroje

B.3.1 Red team

Úkolem Red teamu je zjistit přihlašovací údaje uživatele přihlašujícího se na zabezpečenou stránku. Red team má přístup pouze k VM útočníka, odkud bude realizován útok sslstrip. Veškerá práce bude probíhat pomocí terminálu. Práce bude provedena pomocí *bettercapu*, ve kterém budou nastaveny potřebné úkony pro provedení útoku [113]. *Bettercap* je nástroj pro testování bezpečnosti sítě [114]. Nabízí širokou škálu možností jako je realizace různých typů útoků MITM, odposlechnutí přihlašovacích údajů, manipulace s HTTP, HTTPS, TCP pakety v reálném čase a mnoho jiného [115].

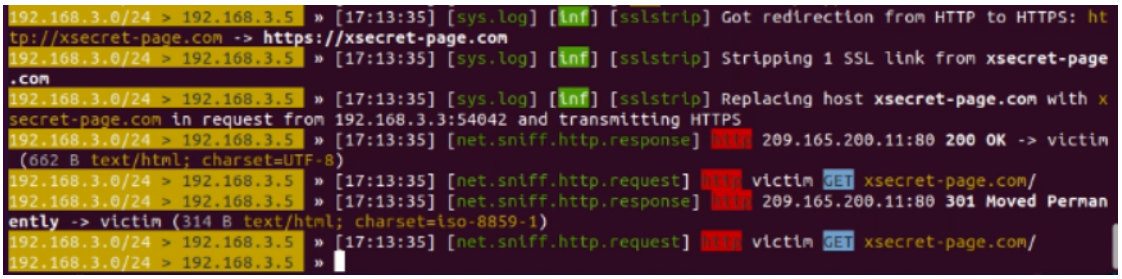
Prvním krokem bude provedení testovacího *pingu* na stanici oběti, kvůli eliminaci přesměrování ze strany routeru. V případě neprovedení *pingu* by pakety byly přesměrovány přes router a nebylo by možné uskutečnit *arpspoof*. Dále si zjistíme pomocí *ifconfig*, které rozhraní je používáno pro komunikaci v síti 192.168.3.0/24. Následně bude přistoupeno k samotnému provedení útoku. Pomocí příkazu *sudo -i* se přepneme do role roota, kde spustíme *bettercap* na daném rozhraní: *bettercap -iface <nazev_rozhrani>*. Po spuštění se objeví síť, ve které se bude naslouchat a kam budou příkazy pro útok zadávány. Následující výpis obsahuje jednotlivé příkazy pro nastavení a spuštění útoku.

```
root@attacker:~# bettercap -iface <nazev_rozhrani>
192.168.3.0/24 > 192.168.3.5 >> set http.proxy.sslstrip true
192.168.3.0/24 > 192.168.3.5 >> set net.sniff.verbose false
192.168.3.0/24 > 192.168.3.5 >> arp.spoof on
192.168.3.0/24 > 192.168.3.5 >> http.proxy on
192.168.3.0/24 > 192.168.3.5 >> net.sniff on
```

Výpis B.1: Nastavení a spuštění útoku ssl-strip

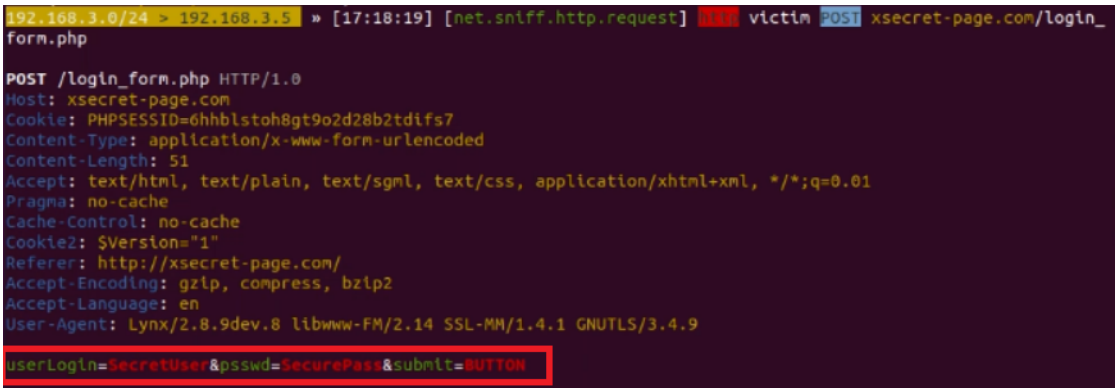
Na začátku se nastaví několik proměnných, které jsou využity v průběhu útoku. První řádek povoluje *http.proxy server*, přes který bude procházet přesměrovaný přenos a který bude realizovat degradaci přenosu z HTTPS na HTTP. Druhý řádek nastavuje zachytávání přenosu a zobrazování pouze relevantních dat, jako jsou zdrojová adresa, cílová adresa nebo hlavička zachyceného paketu.
Následuje zapnutí ARPspoofingu, zapnutí HTTP proxy, zachytávání a zobrazování paketů.

V průběhu zadávání příkazů budou vypsány hlášky týkajících se těchto instrukcí. Například po zapnutí ARPspoofingu bude následovat výpis: *arp.spoof enabling forwarding* a *arp.spoof starting net.recon as a requirement for arp.spoof* a *arp spoofer started, probing 256 targets.*. V případě zapnutí *http.proxy* se vypíše hláška: *started on 192.168.3.5:8080 (sslstrip enabled).*
V tomto kroku již stačí pouze zachytit redirect na požadovanou webovou stránku, který bude použit pro degradaci HTTPS a následně odposlechnout přihlašovací údaje uživatele. Následující obrázek Obr. B.2 ukazuje proces zachycení redirectu ze stránky *http://xsecret-page.com*. Dále následuje degradace HTTPS na HTTP a komunikace mezi serverem a obětí v HTTP.



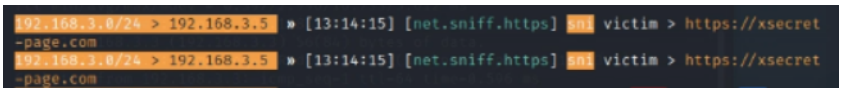
Obr. B.2: Attacker - zachycení redirectu ze stránky xsecret-page.com

Nic netušící oběť zadá své přihlašovací údaje na webové stránce, které se pak odešlou v nešifrované podobě na stroj útočníka, viz Obr. B.3, kde jsou přihlašovací údaje označeny červeným obdélníkem. V opačném případě, když webový server má



Obr. B.3: Attacker - zachycení přihlašovacích údajů ze stránky xsecret-page.com

zapnutou funkci HSTS, tj. trvalý přenos přes HTTPS, útočník není schopen odchytit přihlašovací údaje za realizace stejného postupu. Je pouze schopen zachytit přenesené pakety HTTPS, které jsou přesměrovány přes něho, Obr. B.4.



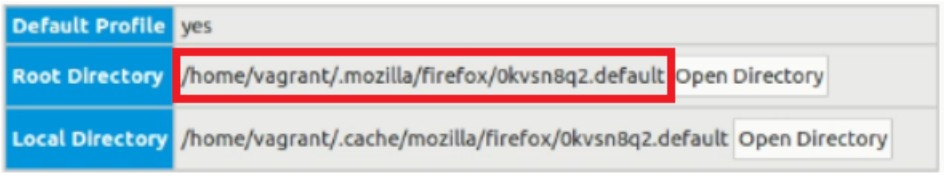
Obr. B.4: Attacker - zachycení HTTPS paketů ze stránky

B.3.2 Blue team

Úkolem modrého týmu bude zabránění dalšího možného útoku, tj. neumožnění komunikace pomocí nezabezpečeného spojení HTTP. Část práce bude probíhat na stanici **victim**, kde se provede prvotní přihlášení, kvůli celkovému odzkoušení funkčnosti. Další část práce bude probíhat na stroji **apache2**, kde bude nastaven protokol HSTS, který pro spojení explicitně vyžaduje zabezpečené připojení. Tedy neumožňuje redirect z portu 80 na port 443.

Nastavení a otestování webové stránky

Na stanici **victim** je nutné stáhnutí certifikátu ze serveru a importování do prohlížeče firefox. Prvně musíme získat výchozí úložiště pro certifikáty na prohlížeči. Spustíme firefox a zadáme dotaz *about:profiles*. Zobrazí se stránka popisující výchozí úložiště pro firefox, viz obrázek Obr. B.5, ze které si zkopírujeme celkovou cestu pro *Root Direcotry* pro import certifikátu.



Obr. B.5: Victim - výchozí úložiště pro firefox

Následně spustíme terminál, kde zadáme následující příkazy:

```
root@victim:~# wget https://xsecret-page.com:443/
xsecret-page.com.pem --ca-certificate=xsecret-page.com.pem
--no-check-certificate
```

Výpis B.2: Stažení certifikátu ze serveru bez provedení kontroly

```
root@victim:~# cp ~/xsecret-page.com.pem /etc/ssl/certs/
xsecret-page.com.pem
```

Výpis B.3: Kopírování certifikátu do /etc/ssl/certs

```
root@victim:~# certutil -A -n "xsecret-page.com" -t "TC,,"
-i /etc/ssl/certs/xsecret-page.com.pem
-d sql:<zkopírovaná_cesta_k_výchozímu_úložišti_firefox>
```

Výpis B.4: Přidání certifikátu do databáze firefoxu

```
root@victim:~# certutil -d
sql:<zkopírovaná_cesta_k_výchozímu_úložišti_firefox> -L
```

Výpis B.5: Ověření přidání certifikátu do databáze

```
root@victim:~# killall firefox
```

Výpis B.6: Ukončení všech procesů pro firefox

Certifikát je instalován pomocí nástroje *certutil*, kde parametr *-A* naznačuje použití anonymních údajů. Další parametry nastavují: *-n* hash pro ověření se vytváří z dat, "název_certifikátu_pro_firefox", *-t* označuje Timeout, *-i <cesta_k_certifikátu>* značí import vybraného certifikátu do databáze, *-d sql:<cesta_k_výchozímu_úložišti _firefox>* značí cílovou databázi. [116]

Po nahrání certifikátu můžeme přistoupit k zobrazení webové stránky. Spustíme prohlížeč firefox a zadáme námi hledanou webovou stránku *xsecret-page.com*. Po dotázání se na IP adresu webové stránky u DNS serveru a proběhlém TCP spojení se pošle HTTP dotaz na webový server, na port 80. V dalším je jako odpověď posláno přesměrování na zabezpečenou stránku, port 443. Musíme uvážit, že v tomto případě je na webovém serveru nastaveno pouze přesměrování stránky z HTTP na HTTPS, tj. zasílá se redirect na přesměrování, který je útočník schopen odchytit. Stránka se zobrazí, ale v případě zadávání přihlašovacích údajů nás upozorňuje, že stránka není bezpečná, obrázek Obr. B.6. Pro kontrolu funkčnosti se přihlásíme na stránku (login=SecretUser, password=SecurePass), tímto jsme odeslali přihlašovací údaje útočníkovi.

Nastavení HSTS

V případě eliminace odposlechu je potřeba nastavit protokol HSTS na straně webového serveru. Kde provedeme několik změn v konfiguraci *apache2* a nastavení webové stránky. V roli roota si otevřeme soubor *nano /etc/apache2/conf-enabled/ssl-params .conf*, kde odkomentujeme řádek *Header always set Strict-Transport-Security "max-age=63072000; includeSubDomains; preload"*. V souborech pro port 80 a 443 přidáme stejný řádek hned za *<VirtualHost *:<číslo_portu>*, Soubory se nachází ve složce */etc/apache2/sites-enabled/*. Jedná se o soubory *000-default.conf* a *default-ssl.conf*. Po nastavení restartujeme *apache2*: *systemctl restart apache2*.

Na stroji oběti vymažeme údaje o prohlížení. Zobrazíme si historii pomocí Ctrl+H. Klikneme na Today, kde se zobrazí již navštívené stránky. Pravým tlačítkem klikneme na odkaz a vybereme poslední možnost: *Forget About This Site* a prohlížeč

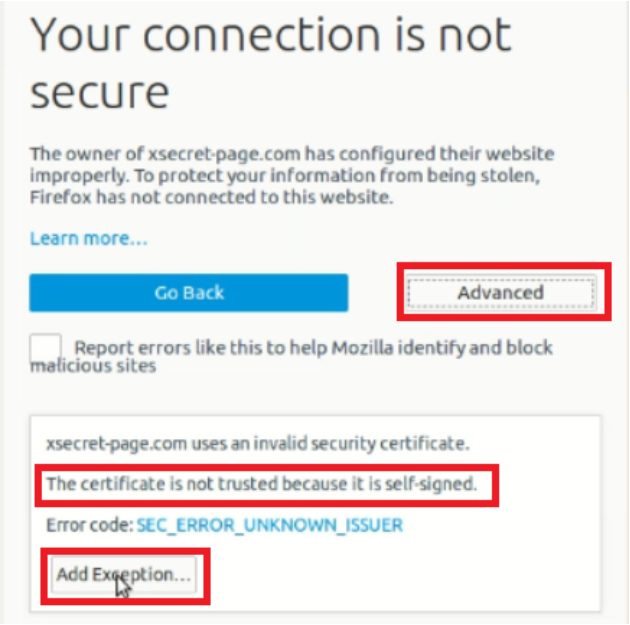
Obr. B.6: Victim - kompromitovaná webová stránka

ukončíme.

Firefox otevřeme znova a zadáme námi hledanou stránku *xsecret-page.com*. Může se stát, že se stránka načte znova pomocí HTTP z důvodu, že prohlížeč přistupuje na tuto stránku poprvé a zatím není informován o zapnutém HSTS. Po otevření stejné stránky na nové kartě se už stránka zobrazí pomocí HTTPS, protože prohlížeč již obdržel informaci vyžadující pouze zabezpečené připojení. Konkrétně se zobrazí hláška ohledně nebezpečného připojení a tím blokuje načtení stránky, viz obrázek Obr. B.7. Nabízí možnost nepřipojení nebo rozšířené, *Advanced*. Víme, že máme nastaven protokol HSTS, který vyžaduje pouze bezpečné připojení, tj. nutné používat certifikát stránky. Vybereme *Add Exception...* a povolíme použití certifikátu stránky. Následně se stránka načte pomocí protokolu HTTPS. Nastavení HTTPS můžeme ověřit i na nové kartě se stejným dotazem. Stránka je automaticky přesměrována na HTTPS, útočník již není schopen odposlechnout přihlašovací údaje. (Pro eliminaci chyby při přihlášení se na nezabezpečenou stránku můžeme změnit přihlašovací údaje.)

Poznámka:

Hláška také obsahuje text *The certificate is not trusted because it is self-signed.*. Jedná se o upozornění, že certifikát není ověřen certifikační autoritou a může jít o podvrh. V tomto případě tomu tak není, náš webový server využívá právě self-signed certificate z testovacích důvodů. V případě ověřeného certifikátu třetí stranou by se stránka ihned měla načíst v zabezpečené formě. Firefox neověřené certifikáty neuznává, tím pádem je musíme nastavit ručně. V případě ověření nastavení HSTS na webovém serveru můžeme využít terminál a zadat příkaz: *curl -I xsecret-page.com* pro ověření nastaveného HSTS protokolu, obrázek Obr. B.8.



Obr. B.7: Victim - vynucení HTTPS

```
HTTP/1.1 301 Moved Permanently
Date: Wed, 03 Feb 2021 17:01:53 GMT
Server: Apache/2.4.18 (Ubuntu)
Strict-Transport-Security: max-age=63072000; includeSubdomains; preload
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Location: https://xsecret-page.com/
Content-Type: text/html; charset=iso-8859-1
```

Obr. B.8: Victim - zobrazení nastaveného HSTS

B.4 Závěr

V laboratorní úloze byl na straně červeného týmu odzkoušen útok sslstrip, pomocí nástroje *bettercap*. SSLstrip úspěšně degradoval HTTPS přenos na HTTP a tím se podařilo získat přihlašovací údaje na webovou stránku. Oproti tomu po nastavení protokolu HSTS se žádné informace získat nepodařilo.
Na straně blue týmu byly zadány přihlašovací údaje z důvodů otestování funkčnosti útoku. Následně na straně webového serveru bylo nastaveno protiopatření, protokol HSTS, které již odchycení přihlašovacích údajů neumožnilo.

C Obsah přílohy

Příloha obsahuje bakalářskou práci v PDF formátu (*Bakalarska_prace.pdf*), dále obsahuje vytvořené sandboxy pro jednotlivé laboratorní úlohy: složka *ddos* a složka *sslstrip*. Každá ze složek obsahuje: složku *base_provisioning* obsahující základní konfiguraci virtuálních strojů, *provisioning* obsahující dodatečnou konfiguraci pro virtuální stroje, *sandbox.yml* s výchozími parametry pro sandbox a *Vagrantfile* pro nástroj Vagrant s parametry pro virtuální stroje. Dále je připojen textový soubor *READ_ME.txt*, který popisuje spuštění sandboxu. Jako poslední je přiložena stručná video ukázka, *ukazka.mp4*, z realizace laboratorní úlohy A.